SONDERHEFT 4/1986 öS 100,- sfr. 14,- Lit. 12000 hfl. 18,- dkr. 58,- DM 14,-

3. SCHNEIDER SONDERHEFT

HAPPY-COMPUTER

Markt&Technik

DAS GROSSE HEIMCOMPUTER-MAGAZIN



Hilfe für Einsteiger

★ Basic leicht gemacht
★ Window-Programmierung
★ Das Innenleben des CPC

CP/M einfach lernen:

Kurs mit vielen Beispielen

Fix gelötet:

8. Bit für Schneider 664/6128

Zum Abtippen

★ Flugsimulator
★ Schach-Tutor
★ Basic-Erweiterung
und jede Menge Listings, Tips und Tricks

Alle Programme auf Diskette und Kassette erhältlich

Leistungsfähige Programmiersprachen für Schneider CPC 464/664/6128 + Joyce

Pascal/MT+

Markt&Technik
Schneider CPC-Software

DIGITAL RESEARCH
Pascal MT+™

FROM DIGITAL RESEARCH™ THE CREATORS OF CP/M

3"-Diskette für CPC 464*, 664*, 6128,
Joyce PCW 8256
Handbuch in englischer Sprache

*Mit 64 KByte Speichererweiterung

Eine der umfangreichsten Pascal-Implementationen für 8-Bit-Mikrocomputer.
Pascal/MT+ ist ein volles ISO-Standard-Pascal, das um eine leistungsfähige Programmierumgebung für Industrie-, Geschäfts- und Ausbildungs-Einsatz sowie Möglichkeiten zur Systemprogrammierung erweitert wurde.
Pascal/MT+ erweitert die bekannten Vorteile der strukturierten Sprache Pascal. Es ist schneller, vielseitiger, portabler und in anspruchsvollen Anwendungen, die die Entwicklung separater Programm-Module erfordern, einfacher zu verwenden.

Direkte Umsetzung in schnellen Objekt-Code

Im Unterschied zu Compilern, die in einen Zwischencode übersetzen, wandelt Pascal/MT+ direkt in schnellen Objekt-Code um. Die Ausführungszeiten sind deshalb wesentlich besser als bei traditionellen Pseudo-Code-Compilern.

Das Pascal/MT+-Paket beinhaltet:

- einen Compiler, der relokatierbare Objekt-Dateien erzeugt,
- einen Linker, der lauffähige Programme erzeugt,
- eine Laufzeitbibliothek
- einen Disassembler, der die Untersuchung des erzeugten Codes ermöglicht, und
- einen Debugger, der einen symbolischen Test eines Programms erlaubt

Die Bibliothek enthält Routinen von der Berechnung transzendenter Funktionen bis zur Verwendung von Maschinen-Interrupten.

Ideal für Geschäfts-, Industrie- und Ausbildungs-Einsatz

Zusätzlich zu den numerischen Standard-Datentypen unterstützt Pascal/MT+ entweder Fließkommazahlen oder binär-codierte Dezimalzahlen (BCD) und erzielt damit die in kommerziellen Anwendungen so wichtige Genauigkeit von Ergebnissen ohne Rundungsfehler.
Für industrielle Anwendungen bietet Pascal/MT+ den Vorteil von ROM-fähigem Maschinencode, Möglichkeiten zur Reduzierung der Programmgröße und erweiterte Ein-/Ausgabefähigkeiten.
Für den Einsatz im Ausbildungs- und Lehrbereich empfiehlt sich Pascal/MT+, weil es eine volle Implementation von Pascal ist, die man leicht erlernen kann, aber später auch bei gestiegenen Ansprüchen noch leistungsfähig ist.

Genügt professionellen Ansprüchen

Pascal/MT+ wurde für die hohen Ansprüche professioneller Softwareentwickler und erfahrener Anwender entwickelt. Pascal/MT+ wird mit ausführlicher Dokumentation in englischer Sprache geliefert.

Hardwarevoraussetzungen

Pascal/MT+ läuft auf den Schneider-Computern CPC 464 und CPC 664 (mit Speichererweiterung), dem CPC 6128 und dem PCW 8256 (Joyce) unter CP/M und CP/M-Plus. Kompilierte Programme sind, bei entsprechender Größe, auch auf dem CPC 464 und CPC 664 ohne Speichererweiterung lauffähig.

Die Vorteile von Pascal/MT+ auf einen Blick:

- Superset des ISO-Standard-Pascal
- Kompilierung separater Module
- erzeugt effektiven Maschinencode
- komplette Entwicklungstools
- erweiterte Datentypen (BYTE, WORD, LONGINT, STRING)
- Bit- und Byte-Manipulationen
- schneller Dateizugriff
- Direktzugriffsdateien
- CHAINing mit Übergabe von Variablen zwischen Overlays
- umfangreiche Dienstprogramme

Best.-Nr. MS 611



CBASIC-Compiler





FROM DIGITAL RESEARCH™ THE CREATORS OF CP/M

3"-Diskette für CPC 464, 664, 6128,
Joyce PCW 8256
Handbuch in englischer Sprache

Der Hochleistungs-BASIC-Compiler für Softwareprofis zur Erstellung kommerzieller Anwendungen.

Der CBASIC-Compiler ist ein erweitertes BASIC mit wichtigen Vorteilen für Softwareprofis. Er ist ein Compiler, der Maschinencode erzeugt und die Programmierung und den Test separater Module erlaubt, die später ein komplettes Programm ergeben sollen. Die integrierten Grafikmöglichkeiten des CBASIC-Compilers erlauben die Programmierung vielseitiger Grafikprogramme für eine Vielzahl von Anwendungen (nur auf Computern mit GSX-Software).

Schnelle Ausführung

Der CBASIC-Compiler kombiniert die Geschwindigkeit von Maschinencode mit der leichten Verständlichkeit der Sprache BASIC. Ein mit dem CBASIC-Compiler kompiliertes Programm wird acht- bis zehnmal schneller ausgeführt als das gleiche interpretierte Programm.

Grafikerweiterungen

Der CBASIC-Compiler beinhaltet einen voll integrierten Satz von Grafikbefehlen und -funktionen. Geräteunabhängige Grafikfähigkeiten ermöglichen die Ausgabe von Grafiken auf jedem unterstützten Grafikausgabegerät (Bildschirm, Drucker, Plotter) ohne Neukompilierung eines Programms.

Dezimal-Arithmetik

Die 14stellige Dezimal-Arithmetik gewährleistet höchste Genauigkeit bei Berechnungen und stellt sicher, daß alle Geldbeträge auf den Pfennig genau stimmen. Rundungsfehler, wie sie bei binärer Arithmetik möglich sind, können nicht auftreten.
Der CBASIC-Compiler unterstützt auch echte Integer-Arithmetik, so daß zur Erhöhung der Geschwindigkeit auch Integer-Variablen verwendet werden können.

Mehrzeilige Funktionen

Durch die Möglichkeit, mehrzeilige Funktionen zu erstellen, verfügt der CBASIC-Compiler über Fähigkeiten, die sich sonst nur in strukturierten Programmiersprachen wie PL/I oder Pascal finden. Innerhalb einer mehrzeiligen Funktion können lokale Variablen verwendet werden.

Für professionellen Einsatz

Der CBASIC-Compiler wurde für die hohen Ansprüche professioneller Softwareentwickler und erfahrener Anwender entwickelt. Der CBASIC-Compiler wird mit ausführlicher Dokumentation in englischer Sprache geliefert.

Hardwarevoraussetzungen

Der CBASIC-Compiler läuft auf Schneider CPC 464 mit Diskettenlaufwerk DDI-1, dem CPC 664, dem CPC 6128 und dem 8256 (Joyce). Für Grafikprogramme wird die GSX-Software benötigt, die nur mit dem CPC 6128 und PCW 8256 (Joyce) ausgeliefert wird. Die Grafiken können dann auf dem Bildschirm oder einem von GSX unterstützten Drucker oder Plotter ausgegeben werden. Es können zum Beispiel der NLQ 401-Matrixdrucker, ein Epson- oder kompatibler Drucker und HP- und HP-kompatible Plotter zur Ausgabe verwendet werden.

Die Vorteile des CBASIC-Compilers auf einen Blick:

- hohe Geschwindigkeit der erzeugten Programme
- Grafikerweiterungen
- Dezimal-Arithmetik mit hoher Genauigkeit
- umfangreiche Stringverarbeitung
- Stringlänge bis 32 KByte
- mehrzeilige Funktionen
- keine Zeilennummern erforderlich
- Overlays durch CHAIN-Befehl

Best.-Nr. MS 612

DM 174,–* (sFr. 158,–/öS 1680,–*)

* inkl. MwSt. Unverbindliche Preisempfehlung



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Diese Markt&Technik-Softwareprodukte erhalten Sie in den Fachabteilungen der Kaufhäuser und in Computershops.

Bestellungen im Ausland bitte an untenstehende Adressen.
Schweiz: Markt&Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042/41 56 56
Österreich: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Straße 24, A-1091 Wien, 02 22/48 15 38-0



Guter Rat ist nicht teuer

Vor zwei Jahren kannte noch niemand den Namen Schneider in der Computer-Branche. Dann kam der CPC 464 und seitdem steht Schneider für preiswerte Geräte mit exellentem Basic, guter Hardware und einem unübertroffenen Gesamtkonzept. Aber nicht nur die Hardware ist auch für den kleinen Geldbeutel geeignet. Fast die gesamte Software für Schneider hat sich auf einem Preisniveau eingependelt, das bedeutend niedriger ist als für andere Computer. Man denke nur an die oft erwähnten CP/M-Programme wie Wordstar, dBase II und Multiplan. Dank Schneider ist die ganze Preisfront ins Wanken geraten. Für uns, die Computer-Besitzer, kann das nur gut sein.

Computer, die mehr als nur das eigene Betriebssystem beherrschen, können auf erheblich mehr Software-Produkte zurückgreifen. Zwei Betriebssysteme (CP/M und Amsdos) verlangen von dem Benutzer aber auch mehr Wissen. Das softwareorientierte Konzept von Schneider erlaubt Eingriffe tief im Inneren der Firmware. Um Ihr Wissen zu erweitern, halten Sie mit unserem dritten Sonderheft wieder mehr als 130 Seiten preiswerte Informationen in der Hand.

Wer seinen Schneider erst neu gekauft hat, findet im großen Einsteigerteil alles nötige für den erfolgreichen Anfang, auch Antworten auf Fragen, die in keinem Handbuch beantwortet werden: Wie Sie in Basic programmieren können, welche trickreichen Befehle es unter dem Schneider-Basic gibt und wie Sie am besten Ihre Daten sichern, sowie vieles mehr. Manches davon ist auch für Profis wissenswert.

CP/M heißt ein Verkaufsargument der Leute aus Türkheim. Bloß allein das Wort hilft niemandem. Massenhaft Programme soll es unter diesem Betriebssystem geben – aber das kann doch noch nicht alles sein. Richtig! Was es mit diesem neuen alten Standard auf sich hat, das lernen Sie in unserem Kurs. Auch wie Sie unter CP/M Dinge machen können, die Sie eigentlich nicht machen dürfen, wird Ihnen gezeigt. Denn die Firmware Ihres Schneiders läßt sich natürlich auch unter CP/M nutzen.

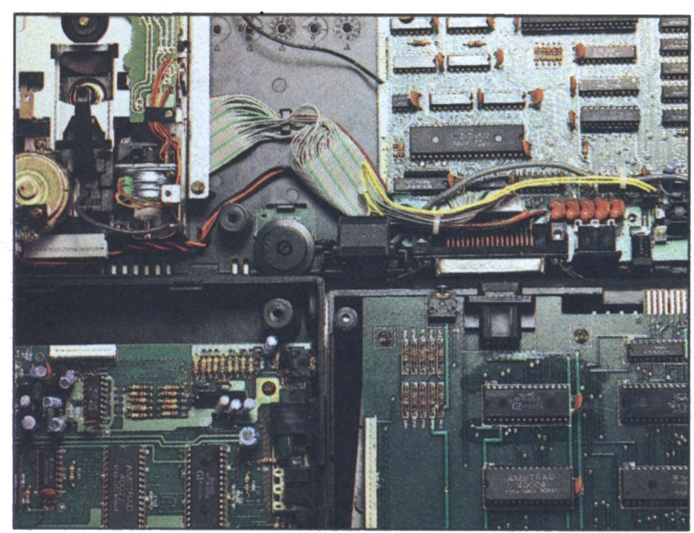
Das letzte Sonderheft sei zwar sehr gut und informativ gewesen, haben viele unserer Leser geschrieben. Aber ein paar Spiele zum Abtippen, die hätten doch gefehlt. Wir haben uns das zu Herzen genommen und deshalb den Anteil der Spiele-Listings ganz gewaltig aufgestockt. Steuern Sie als Pilot Ihren Schneider über Deutschlands Luftstraßen. Oder experimentieren Sie mit einer mathematischen Spielerei: »Leben« simuliert die Entwicklungsgeschichte. Tips & Tricks sowie Software für »ernste Zwecke« ist natürlich auch vorhanden. Und alle Programme gibt es wieder auf Kassette oder Diskette.

Daß es auch in nächster Zeit an der »Schneider-Front« nicht ruhig werden wird, ist abzusehen. Vortex will jetzt endlich seine schon lang angekündigte MS-DOS-Karte auf den Markt bringen. Und für den CPC 6128 soll eine Superspeicherkarte den RAM-Bereich auf über 1 MByte aufmotzen. Daß dabei auch gleich ein neuer Prozessor und verschiedene Schnittstellen eingebaut werden, ist fast schon nebensächlich.

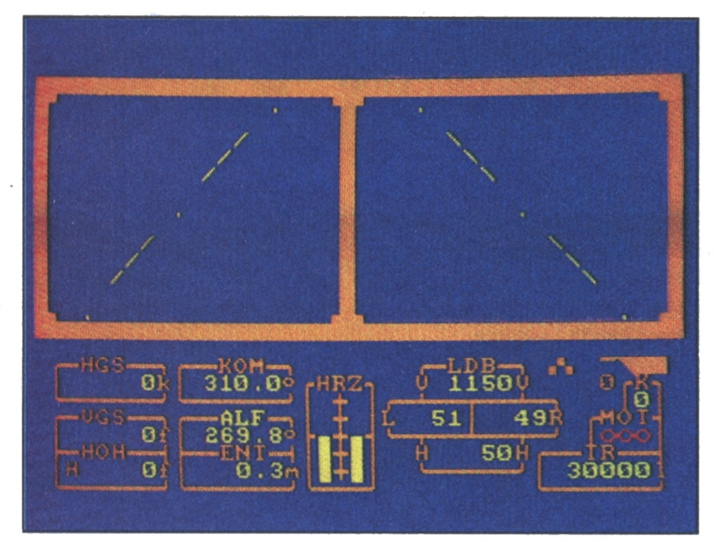
Und auch der Buchverlag in unserem eigenen Haus hat interessante Produkte auf Lager. So wird die gesamte Literatur von Digital Research über CP/M-2.2 angeboten. Und auch das oft geschmähte Logo wird mit Büchern »umworben«. Die Programmpalette der preiswerten CP/M-Software wird von Markt&Technik ebenfalls erweitert. Auch andere Verlage bauen gute Programme vom Commodore 64 auf die Schneider-Computer um. Wo man hinschaut: An allen Ecken und Enden wächst was für den Schneider. Da kann unser Hobby doch nicht langweilig werden!

Andreas Hagedorn

Inhalt



Auf die Anatomie der Schneider-Computer gehen unsere Grundlagenbeiträge in diesem Schneider-Sonderheft ein. 14



In ganz Deutschland können Sie mit unserem Flugsimulator auf jedem größeren Flughafen landen: »Bitte anschnallen und das Rauchen einstellen.« 82

Programm-Service

Wer keine Zeit oder keine Lust hat, alle Programme selbst in mühevoller Kleinarbeit abzutippen, kann wieder auf den bewährten Programm-Service zurückgreifen. Es sind hier sämtliche Programme des Sonderhefts auf einer Diskette oder Kassette erhältlich.

Bestellnummer LH 86S4 D 34,90 DM*

Bestellnummer LH 86S4 K 29,90 DM*

(*inkl. MwSt.)

Spieletest

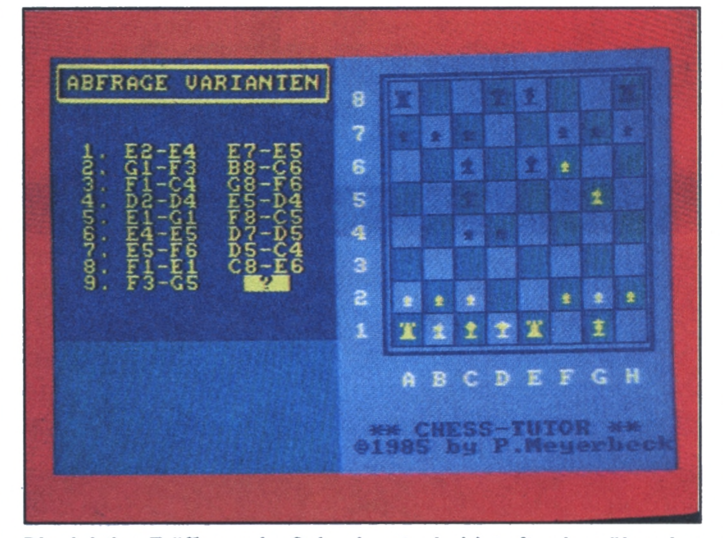
Bastelei

Eingabehilfe

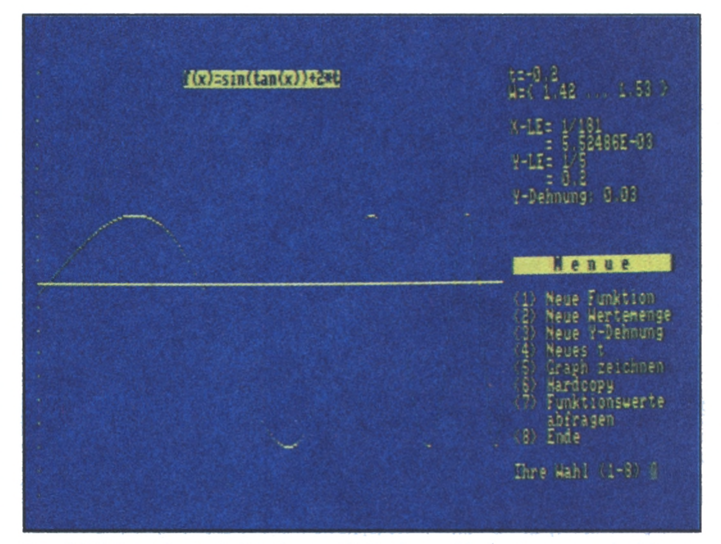
Grundlagen

Basic

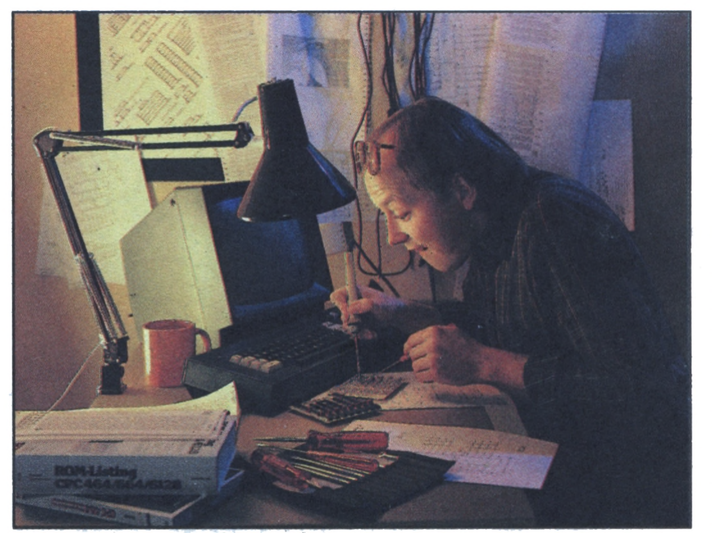
Grafik



Die richtige Eröffnung im Schach entscheidet oft schon über den Ausgang des Spiels. Ob sizilianisch oder spanisch – unser Schach-Trainer hat für jeden etwas. 102



Der CPC fungiert als Mathematik-Trainer. Sie bekommen in kürzester Zeit einen Blick dafür, wie zum Beispiel die zugehörigen Graphen zu eingehenden Funktionen auszusehen haben. 107

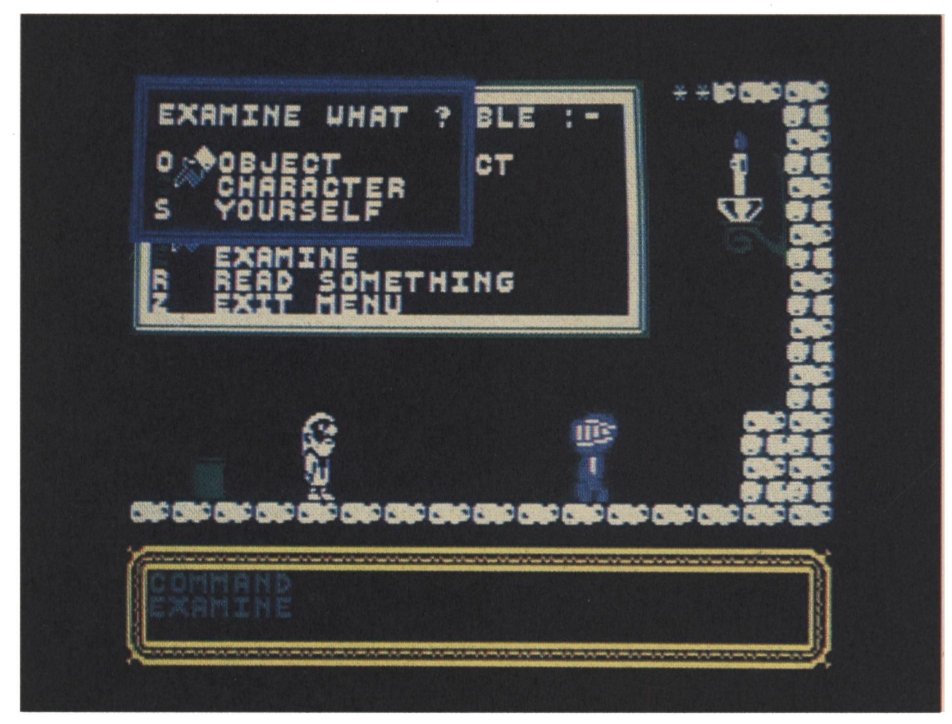


Wer basteln kann, hat mehr von seinem Computer. Wir zeigen, wie Sie sich eine echte Centronics-Schnittstelle für Ihren CPC 664 oder 6128 bauen können. 10

Schneider spielt weiter



Der Schneider CPC hat beileibe nicht »ausgespielt«; ganz im Gegenteil: Als Spiel-Maschine erfreut er sich immer größerer Beachtung der britischen Softwarefirmen. Brillante Billig-Spiele, gute Umsetzungen und Neuerscheinungen stellen wir Ihnen vor.



Das windowreiche Action-Adventure »Spellbound«

In diesem Artikel wollen wir ganz unten anfangen – nämlich ganz unten in der Preis-Skala. Im Preisbereich zwischen 10 und 20 Mark sind jetzt auch für Schneider CPCs einige rundum empfehlenswerte »Billig-Spiele« erschienen, bei denen nicht an der Qualität gespart wurde.

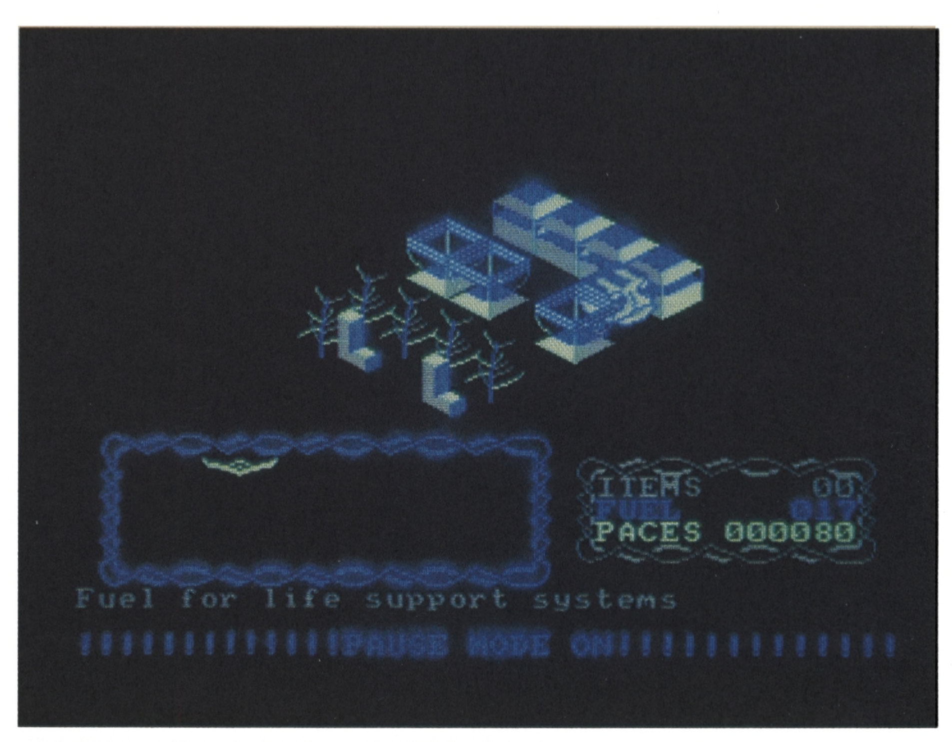
Der größte Anbieter von Software-Preisbrechern, Mastertronic, hat in seiner neuen »MAD-Games«-Reihe jetzt auch einen Schneider-Titel. »Spellbound« ist eine Umsetzung des Spectrum-Originals, das beim Test in jeder Hinsicht das blanke Entzücken hervorrief. Für 15 Mark erhält man ein spielerisch sehr ausgefeiltes Action-Adventure, bei dem Sie in der Rolle des Magic Knight (richtig, der aus »Finder's Keepers«) im Schloß von Karn herumflitzen und Ihren Zauberboß Gimbal befreien müssen. Durch klug aufgebaute Untermenüs, die als höchst attraktive Pracht-Windows auf dem Bildschirm erscheinen, können Sie Gegenstände aufsammeln und untersuchen, andere Spielfiguren beklauen, deren Charakter begutachten (ist's ein Guter oder ein Böser?) und vieles mehr.

Ganz ohne Joystick (beziehungsweise Tastatur) kommt man bei »Spellbound« nicht über die Runden, aber der spielerische Schwerpunkt liegt bei der Abenteuer-Komponente: Nachdenken und ausprobieren heißt die Parole. Ein sehr ansprechendes Spiel, einfach zu bedienen, grafisch gelungen und ausgesprochen preisgünstig – somit ein ganz heißer Kauf-Tip.

In derselben Preisklasse liegt »Cylu«, ein Action-Adventure aus Firebird's »Super Silver Range«. In der Rolle des Titelhelden müssen Sie in einem riesigen Labyrinth mit recht schmucker perspektivischer Grafik 24 Gegenstände finden, zum Startpunkt zurückkehren, und damit den völlig aufgelösten »Master Computer« reparieren (Das hat man nun von der Technik!). Sperren, Teleporter-Felder und ähnliche Spezialitäten des Hauses erschweren die Labyrinth-Suche. Außerdem müssen Sie immer wieder Ihren dahinschwindenden Sprit-Vorrat ergänzen und können nur fünf Gegenstände gleichzeitig bei sich tragen.

Auch dieses Programm ist eine geschickte Mischung aus Geschicklichkeits- und Abenteuer-Spiel. Durch die etwas langsame Grafik ist es weniger für Action-Enthusiasten geeignet. Grafisch und spielerisch erinnert es allerdings sehr an die teureren Erfolgstitel »Knight Lore« und »Alien 8«. Vor Doppelkäufen sei gewarnt!

Aus demselben Hause stammt das »Willow Pattern Adventure«. Mit einem richtigen Adventure hat das Programm



»Knight Lore«-Verschnitt für wenig Geld: »Cylu«



eigentlich herzlich wenig zu tun; es ist ein unterhaltsamer Vertreter des unerschütterlichen Genres der Geschicklichkeits-Spiele. Die Story ist aus bewährten Zutaten zusammengestrickt: 1. Spieler verkörpert Chang, den jungen Helden. 2. Liebliche Prinzessin wird von Bösewicht entführt. 3. Junger Held zögert nicht und fegt los, um besagte Prinzessin zu retten.

Sie müssen Ihren Chang durch ein Labyrinth steuern, das entfernt an »Sabre Wulf« erinnert. Beim Umherirren wird öfters der Weg von einem schwertschwingenden Samurai versperrt. Wenn Sie vorher einen Säbel aufgesammelt haben, können Sie den Wächter mit einem Druck auf den Feuerknopf ausschalten.

Für knapp 20 Mark erhält man diese nette Mischung aus Hüpf- und Action-Spiel. Vor allem für den preisbewußten Software-Käufer ein empfehlenswertes Programm, das gute Grafik sowie wohlklingenden Sound bietet und nicht allzu schwierig ist.

Ein weiterer empfehlenswerter Vertreter der Abteilung »Viel Spiel fürs Geld« ist die Zusammenstellung »They sold a Million«. Für 39 bis 59 Mark (Kassette und Diskette) erhält man gleich vier etwas ältere, aber immer noch spielenswerte Bestseller-Programme. Es handelt sich um das Geschicklichkeits-Spiel »Jet Set Willy«, den Action-Titel »Sabre Wulf« und die beiden Sport-Simulationen »Daley Thompson's Decathlon« (Leichtathletik) und »Rocco« (Boxen). Vor allem für Einsteiger, die noch keines der vier Spiele haben, ist dieser Vierer-Pack eine lohnende Anschaffung.

Bei den immer noch sehr beliebten Sport-Spielen hat sich in den letzten Wochen eine ganze Menge getan. Activision hat seine Box-Simulation mit dem Überlängen-Titel »Barry McGuigan's World Championship Boxing« (Uff!) vom C 64 für den Schneider umgesetzt. Zur allgemeinen freudigen Überraschung ist diese Adaption sehr gut gelungen.

Sie starten als hoffnungsvoller Newcomer in die rauhe Welt des Profi-Boxens. Sieg für Sieg klettern Sie die Top 20 der Weltrangliste empor, um eines Tages den Titelkampf gegen den britischen Box-Matador Barry McGuigan zu bestreiten. Zu Beginn können Sie sich einen bestimmten Kämpfer-Typ aussuchen und dann im Trainingslager entscheiden, welche Kampfstärke Sie fördern wollen. Acht verschiedene Schlag- und Defensiv-Techniken stehen im Ring zur Verfügung. Sie sind nicht auf den Kampf gegen den Computer beschränkt, sondern können auch gegen einen Freund die Fäuste schwingen.

»World Championship Boxing« ist ein Sportspiel von der anspruchsvollen Sorte. Es kommt auf Technik und Strategie und nicht auf plumpes Draufhauen an. Das Programm hat derart viele Feinheiten, daß es lange Zeit eine Menge Spaß macht. Grafisch ist es eine quasi identische Kopie des C64-Originals. Die Programmierer haben sich für die CPC-Umsetzung die nötige Zeit gelassen, was dem Endprodukt sehr zugute kommt. Wer auch einmal in den Ring steigen will, ist mit 39 bis 59 Mark (Kassette und Diskette) dabei.



Vier Fäuste für den Titelkampf: »World Championship Boxing«

Wenn es Sie mehr nach den fernöstlichen Kampfsportarten gelüstet, dann dürfen Sie bei der Schneider-Version des Spielhallenautomaten »Yie Ar Kung-Fu« die Handkanten wirbeln lassen. Zehn verschiedene Schläge und Bewegungen können Sie ausführen, während der Reihe nach acht grimmige Gegner auf Sie losgelassen werden. Jeder prügelwillige Kontrahent hat eine schurkige Spezialität. Da gibt es einen Burschen, der öfters mal einen Hechtsprung durch die Luft macht, oder den Krieger, der auch mit einem Schwert zuschlägt.



Fernöstliches Labyrinth-Verwirrspiel: »Willow Pattern Adventure«

Um die Motivation (und damit auch den Frust) erheblich zu steigern, kann man die Gegner nicht einzeln anwählen. Wenn Sie sich also bis zum großen Kung-Fu-Meister namens Blues durchkämpfen wollen, müssen Sie mit dem Joystick schon ganz schön gewieft sein. Technisch ist das Spiel nicht so anspruchsvoll wie der Melbourne-House-Renner »Exploding Fist«, doch die unterschiedlichen Gegner und die wirklich schöne Grafik machen »Yie Ar Kung-Fu« sehr interessant. Es ist mehr ein Action- als ein Sport-Spiel und für Freunde von flotten Reaktionstests eine echte Herausforderung.

Richtig hemmungslos geprügelt wird dann bei »Fighting Warrior«, wo Sie eine Prinzessin aus den Fängen des Standard-Bösewichts retten sollen. Besagter Unsympath (diesmal ist's ein fieser Pharao, schließlich befinden wir uns im alten Ägypten) hetzt nun einen greulichen Kämpfer nach dem anderen auf Sie. Da hilft nur eines: Hauen und nicht verhauen werden!

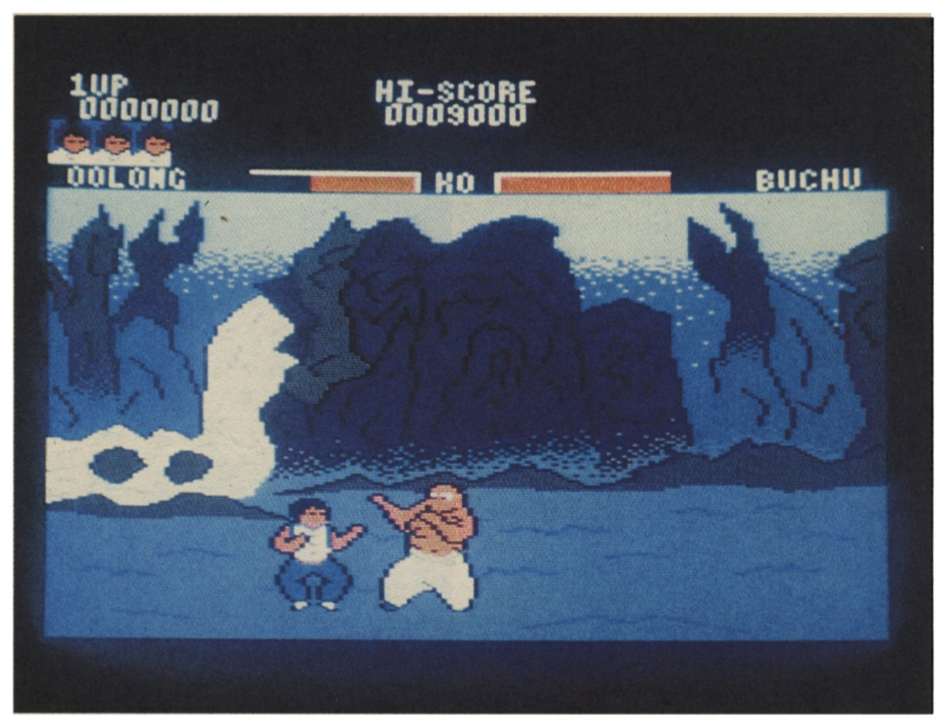
Die Software-Sprites sind groß und gut animiert. Doch der Spielwitz kann nicht ganz mit der Grafik mithalten, denn auf Dauer wird die Klopperei doch etwas öde. Für die Sammler martialischer Kampfsport-Exzesse interessant, aber ansonsten eine etwas eintönige Angelegenheit.

Wenden wir uns friedlicheren Dingen wie der Winterolympiade »Winter Sports« zu. Das ist eines von den Programmen, die nur so lange einen guten Eindruck machen, bis man zu spielen beginnt. Die acht Disziplinen wie Eishockey und Biathlon sind alle schrecklich langsam, grafisch plump, und den Sound sucht man gänzlich vergebens. Selbst wenn Sie Sportspiele heiß und innig lieben, sollten Sie von diesem enttäuschenden Programm die Finger lassen. Da lohnt es sich, auf die Schneider-Version des Epyx-Supersellers »Winter Games« zu warten, die uns zu Redaktionsschluß leider noch nicht vorlag.

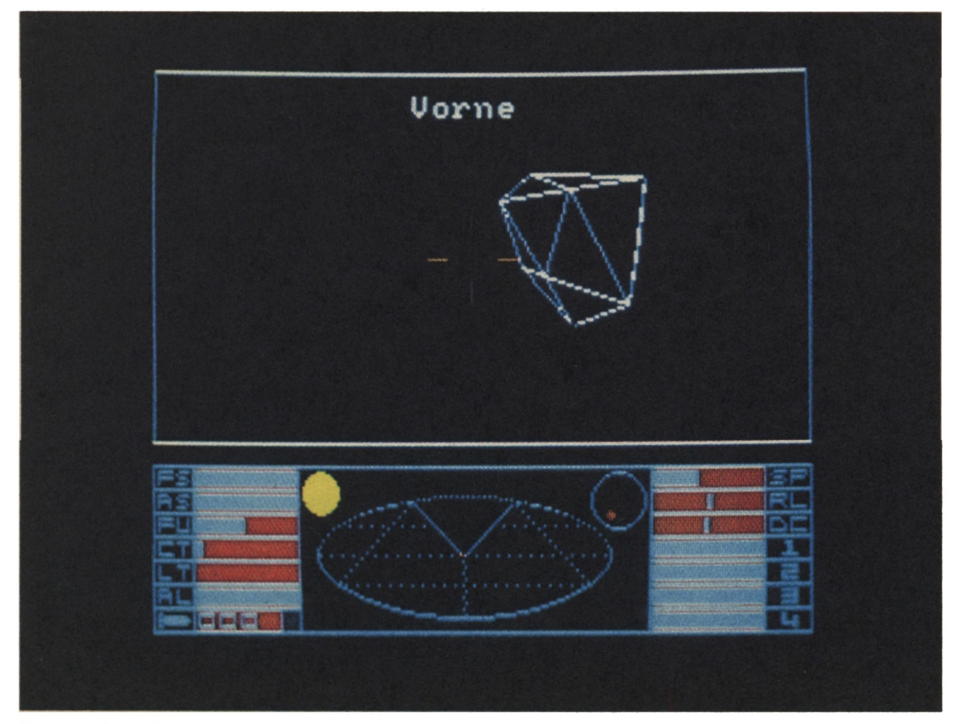
Umsetzungen von C64-Spielen gab es in den letzten Wochen reichlich. Einige Softwarefirmen präsentieren immer noch erschreckend schwache Adaptionen, aber einige Titel beweisen, daß es auch besser geht.

So ist das schwierige Action-Adventure »Hexenküche« von Palace sehr gut gelungen. Das fesselnde Spiel um eine Hexe, die stilecht auf einem Besenstiel durch die Lande düst, gehört grafisch und spielerisch zur Oberklasse. Das vergnügliche Programm ist zudem noch zu recht fairen Preisen erhältlich: Je nach Datenträger (Kassette oder Diskette) kostet es zwischen 29 und 49 Mark.

Eines der Kult-Spiele überhaupt, »Elite«, liegt jetzt auch in der Schneider-Version vor. Die fesselnde Mischung aus Strategie-, Wirtschafts-, Action-Spiel und Flugsimulator stürmte in ganz Europa die Hitlisten und heimste Kritikerlob in rauhen Mengen ein.



Hau' erbarmungslos zu bei »Yie Ar Kung-Fu«



Ein Fest für Langzeit-Spieler: »Elite«

Im Vergleich zur C64-Version sind die spektakulären 3D-Vektorgrafiken zwar langsamer, doch dafür werden mehr Farben verwendet. Inhaltlich blieb alles beim alten. Die Schneider-Version bietet sogar zwei der undurchsichtigen Geheim-Missionen mehr, mit denen Sie auf Ihrem Weg durch acht Galaxien betraut werden können. »Elite« gibt es in einer voll eingedeutschten Version. Außerdem läuft zur Zeit ein großer Wettbewerb, den Happy-Computer zusammen mit Firebird Software ausschreibt.

Bei diesem erfreulichen Angebot müßte auch für Sie der eine oder andere Favorit dabei sein. Beim Kampf gegen Raumpiraten, Mittelgewichtsboxer & Co. wünsche ich Ihnen bis zum nächsten Schneider-Sonderheft viel Vergnügen und bruchsichere Joysticks!

(Heinrich Lenhardt/hg)

PROGRAMM-SERVICE



Bestellungen in der Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Tel. 042/41 56 56
Bestellungen in Österreich: Bücherzentrum Meidling, Schönbrunner Straße 261, A-1120 Wien, Tel. 0222/83 31 96,
Microcomput-ique E. Schiller, Fasangasse 21, A-1030 Wien, Tel. 0222/78 56 61,
Ueberreuter Media Handels- und Verlagsgesellschaft mbH, Alser Straße 24, A-1091 Wien, Tel. 0222/48 15 38-0
Bestellungen aus anderen Ländern bitte per Auslandspostanweisung!

Das Angebot dieser Ausgabe:

Anwendungs-Programme:

Chesstutor. Üben Sie Schach-Eröffnungen. **Funktionsplot** (nur CPC 464). Mathematische Funktions-Graphen auf Bildschirm und Papier. **Division.** Unentbehrlich für exakte Berechnungen von Brüchen. **Dateiverwaltung.** Der komfortable Notizblock.

Spiele:

Faszination Leben. Mit »Life« dem Ursprung auf der Spur. **Jetliner.** Ein fantastischer Flugsimulator für alle CPCs. **Chopper** (nur CPC 464). Erkunden Sie die Höhlen des schrecklichen Dr. Argh. **Schnellboot-Kommandant.** Gefechte auf hoher See für echte Kämpfernaturen. **Light-Cycles** (nur CPC 464). Nur taktisches Geschick und Reaktionsvermögen führen zum Ziel. Hohe Geschwindigkeit durch 100 Prozent Maschinencode. Plus sämtlicher Tips & Tricks-Listings.

Kassette, Best.-Nr. LH 86S4 K
DM 29,90* sFr. 24,90/öS 299,–*

Diskette 3 Zoll, Best.-Nr. LH 86S4 D
DM 34,90* sFr. 29,50/öS 349,–*

Programme aus früheren Ausgaben:

Happy-Computer, Ausgabe 4/86
Schneider CPC
D-Mon.
Daten auf Diskette Byte für Byte lesen und ändern. Fehlerhafte Dateien korrigieren und retten.
GOTO XY (nur CPC 464).
Eine mächtige RSX-Befehlserweiterung, die erlaubt, das Ziel von GOTO-GOSUB-Befehlen mit Hilfe einer Variablen zu bestimmen.
Accept.
Ein komfortabler Ersatz für den normalen INPUT-Befehl, mit dem sich jetzt die maximale Eingabe-Länge begrenzen läßt.
Turbo-Screen (nur CPC 464).
Mit dieser RSX-Erweiterung machen Sie der Bildschirmausgabe im Modus 2 Beine.
Aus Ausgabe 2/86.
Explora.
Mit diesem Prüfsummen-Generator entfällt die lästige und zeitaufwendige Fehlersuche.
Stack-Manipulation (nur CPC 464).
Basic-Programmierung mit vier RSX-Befehlen.
Aus Ausgabe 3/86.
Tool-Basic.
44 neue RSX-Befehle für Grafik-, Sprite-, Disketten- und Kassetten-Programmierung.
Achtes Bit.
Endlich Abhilfe für den Umstand, daß der Schneider CPC über die Drucker-Schnittstelle nur sieben Datenbits ausgibt.
Mord im Computer.
Das DFÜ-Spiel mit Adventure-Charakter.
Aus Ausgabe 4/86.
Best.-Nr. LH 8604 SK (Kassette)
DM 29,90*/sFr. 24,90/öS 299,–*
Best.-Nr. LH 8604 SD (Diskette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 3/86
Commodore 64/Commodore 128
Copter-Fight
Ein interessantes Hubschrauber-Kampfspiel für zwei Personen. Geschicklichkeit und Reaktionsvermögen sind Trumpf.
Husky-Basic
Die mächtige Basic-Erweiterung für Grafik, Sound und strukturiertes Programmieren.
Unser Sonnensystem
Lernen Sie mit diesem Anwendungs-Programm alle Planeten unseres Sonnensystems kennen (mit Simons Basic).
Wahlautomat
Hardware-Bastelei. Lassen Sie Ihren C 64 Telefonverbindungen anwählen! Zusatz zum Listing des Monats aus HAPPY 2/86.
Softpaint
Ein menügesteuertes Zeichen- und Malprogramm für den Commodore 128 im C 128-Modus (kein C 64-Programm).
Bestell-Nr. LH 8603 CD
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 2/86
Commodore 64
Oval Pattern
Machen Sie die Kurvendiskussion auf dem C 64 interessant und nutzen Sie gleichzeitig die tollen Grafikmöglichkeiten dieses Computers voll aus.
Börse
Lernen Sie das Börsengeschehen spielend kennen. »Börse« simuliert mit Grafik und Text die Abläufe und Vorgänge an der Börse.
Poster Hardcopy
Dieses Programm fertigt auf Ihrem Drucker einen 75 x 56 cm großen Ausdruck des Commodore-64-Grafik-Speichers an.
Kassetten-Designer
Eine hervorragende Hilfe bei der Archivierung von Ihren Computer- oder Musikkassetten.
Super-Sprite
Eine Maschinencode-Routine zur professionellen Sprite-Bewegung. Machen Sie Ihren Commodore zu einem Trickfilm-Generator.
Transbit
Das Listing des Monats ist ein Terminalprogramm der Spitzenklasse für Ihren Commodore 64. Datenfernübertragung ist kein Problem mehr.
Alle 6 Programme auf Diskette für den Commodore 64.
Bestell-Nr. LH 8602 CD
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 1/86
Commodore 64/Commodore 128
Taxi. Aus Ausgabe 1/86.
Musik und Farbe. Aus Ausgabe 12/85.
SDB-Sprite Mover. Aus Ausgabe 1/86.
ES-AE. Aus Ausgabe 1/86.
Ultraload. Aus Ausgabe 1/86.
Error 64. Aus Ausgabe 1/86.
Scroll 64. Aus Ausgabe 1/86.
Schatzsuche. Aus Ausgabe 12/85.
SLAD. Aus Ausgabe 12/84.
Alle 9 Programme auf Diskette für den Commodore 64/128
Bestell-Nr. LH 8601 CD
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 12/85
Atari 800XL/130XE/800
Bestell-Nr. LH 8512 B
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 12/85
Schneider CPC
Programmtransfer leichtgemacht (zwei Programme, S. 72). »Tasword 464« mit DIN-Tastatur
Bewegte Grafik mit drei Befehlen Maschinencode-Routinen in BASIC umgesetzt.
Aus Ausgabe 10/85.
Sam – (fünf Programme, S. 109).
Aus Ausgabe 11/85.
Deutscher Zeichensatz unter CP/M. Hardcopy.
RSX-Befehle mit direkter Stringvariable.
Aus Ausgabe 12/85.
Alle 8 Programme auf einer Kassette oder Diskette für den Schneider CPC.
Bestell-Nr. LH 8512 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,–*
Bestell-Nr. LH 8512 D (Diskette)
DM 34,90*/sFr. 29,50/öS 349,–*

Happy-Computer, Ausgabe 11/85
Commodore 64
Bestell-Nr. LH 8511 A
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 10/85
Sinclair Spectrum
Bestell-Nr. LH 8510 D
DM 19,90*/sFr. 17,–/öS 199,–*
Atari 800XL
Bestell-Nr. LH 8510 B
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 9/85
Commodore 64
Bestell-Nr. LH 8509 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 8/85
Schneider CPC 464
Bestell-Nr. LH 8508 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 7/85
Commodore 64
Bestell-Nr. LH 8507 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 6/85
Commodore 64
Bestell-Nr. LH 8506 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 5/85
Schneider CPC 464
Bestell-Nr. LH 8505 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 4/85
Commodore 64
Bestell-Nr. LH 8504 A (Diskette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Computer, Ausgabe 3/85
Schneider CPC 464
Bestell-Nr. LH 8503 G (Kassette)
DM 29,90*/sFr. 24,90/öS 299,–*

Happy-Sonderhefte

Sonderheft 3/86: 68000
Bestell-Nr. LH 86S3 D (Diskette)
DM 29,90*/sFr. 24,90/öS 299,–*

Sonderheft 2/86: ATARI
Bestell-Nr. LH 86S2 D (2 Disketten)
DM 34,90*/sFr. 29,50/öS 349,–*

Sonderheft 1/86: Schneider
Bestell-Nr. LH 86S1 D (Diskette)
DM 34,90*/sFr. 29,50/öS 349,–*
Bestell-Nr. LH 86S1 K (Kassette)
DM 29,90*/sFr. 24,90/öS 299,–*

Sonderheft 2/85: Schneider
Bestell-Nr. LH 85S2 D (3"-Diskette)
DM 34,90*/sFr. 29,50/öS 349,–*
Bestell-Nr. LH 85S2 V (5½"-Diskette)
DM 34,90*/sFr. 29,50/öS 349,–*
Bestell-Nr. LH 85S2 K (Kassette)
DM 29,90*/sFr. 24,90/öS 299,–*

Sonderheft 1/85: Spectrum
Bestell-Nr. LH 85S1 D (Kassette)
DM 19,90*/sFr. 17,–/öS 199,–*

* inkl. MwSt. Unverbindliche Preisempfehlung

Bitte verwenden Sie für Ihre Bestellung und Überweisung die eingeheftete Postgiro-Zahlkarte, oder senden Sie uns einen Verrechnungs-Scheck mit Ihrer Bestellung. Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

Das achte Bit



Erweitern Sie die Centronics-Schnittstelle Ihres Computers auf acht Bit. Bei allen drei Geräten von Schneider muß jeweils nur ein Draht eingelötet werden. Eine kleine Maschinencode-Routine bindet den neuen Port vollwertig in das Betriebssystem ein.

Für die Besitzer der Schneider CPC 464 gibt es sie schon seit dem Sonderheft 2/85 von Happy-Computer: die vollwertige Centronics-Schnittstelle mit acht Bit. Der Umbau am Computer ist bei den großen Brüdern CPC 664 und 6128 der gleiche – wenn man vom unterschiedlichen Aufbau der Platine absieht. Wieder muß nur eine Drahtbrücke zwischen Pin 12 des 8255-Ein-/Ausgabebausteins und der achten Datenleitung des Druckeranschlusses gelegt werden. Komplizierter ist die Anpassung der Ausgaberoutine, da sich die Startadressen beim 664 (und beim 6128 auch schon wieder) geändert haben.

Die Druckerschnittstelle der Schneider-Computer entspricht bis auf das fehlende achte Bit voll der Centronics-Norm. Damit kann man jeden Drucker, der dieser Norm entspricht (und das sind fast alle Geräte), ohne kompliziertes Interface anschließen. Einfach ein passendes Kabel besorgen, anschließen und schon werden mit »PRINTER # 8« die Zeichen ausgegeben. Einen Fehler hat das Ganze aber noch. Da der Schneider über Pin 14 neben dem normalen Carriage Return (Wagenrücklauf) auch einen Line Feed (Zeilenvorschub) veranlaßt, die meisten Drucker dies aber von Haus aus am Ende jeder Zeile (nach Carriage Return) machen, entsteht zwischen zwei Druckzeilen immer eine Leerzeile. Die ist nur dadurch zu umgehen, daß man Leitung 14 am Centronics-Kabel unterbricht. Spezielle für den Schneider angebotene Druckeranschlußkabel haben diese Unterbrechnung schon eingebaut.

Ist der Drucker installiert, kann man über den normalen Centronics-Port alle Symbole mit einem ASCII-Code unter 127 problemlos ausgeben. Da das achte Bit aber immer 0 ist, können die höherwertigen Zeichen nicht ausgedruckt werden. Einzelnadelansteuerung – zum Beispiel für Hardcopys – ist mit einer »kastrierten« Schnittstelle nur umständlich zu programmieren.

Warum haben die Entwickler bei Amstrad auf dieses achte Bit verzichtet? Der Grund ist einfach und im Konzept der Amstrad-(Schneider-)Computer zu suchen. Um das Gerät so preiswert wie möglich zu halten, wurde diese achte Datenleitung als Strobe-Signal genutzt. Aber mit Port C vom Ein-/Ausgabe-Baustein 8255 kann man diesen Fehler beheben. Normalerweise dient er dem Kassettenrecorderanschluß. Da Drucker und Datenspeicher nie gemeinsam gebraucht werden, kann Port C als 8 Bit zweckentfremdet werden. Die richtige Auswahl der Daten, damit an das »achte Bit« nicht versehentlich Druckerinformationen geschickt werden, wenn Daten gespeichert werden sollen, besorgt unsere Routine (Listing 1).

Verbinden wir jetzt Port C des 8255 mit unserem Centronics-Port und ändern die Ausgabe-Routine im Betriebssystem, so haben wir einen Computer mit vollwertigem Anschluß. Probleme mit der Datenaufzeichnung auf dem Kassettenrecorder gibt es nicht, da der Drucker und das Tonbandgerät nie gleichzeitig angesprochen werden. Eventuelles Datenwirrwarr am Drucker (während der Arbeit mit dem Kassettenrecorder) oder am Kassettenrecorder-

Port (während der Arbeit mit dem Drucker) stört nicht.

Alles was Sie für den Umbau brauchen, ist ein Lötkolben, Draht (4 Zentimeter beim 664 und 10 Zentimeter beim 6128) und etwas handwerkliches Geschick. Zuerst lösen Sie die sechs Schrauben an der Unterseite des Gehäuses und entfernen den Teil mit der Tastatur. Die Stecker können beim späteren Zusammenbau nicht falsch verbunden werden, da sie farblich gekennzeichnet sind und außerdem nur in eine Richtung passen.

Am Centronics-Port zählen Sie nun von links den neunten Platinenfinger ab. Beim 6128 nehmen Sie den neunten Verbindungsdraht zwischen Stecker und Platine (siehe auch Bild 1 und 2). Dieses achte Bit ist beim Schneider auf Masse gelegt und erscheint damit dem Drucker immer als nicht gesetzt.

Beim 664 müssen Sie jetzt die Leiterbahn zu diesem Finger mit einem Leiterbahnunterbrecher (oder einem scharfen Messer) auftrennen. Beim 6128 machen Sie das einfach, indem Sie den Draht mit einer Zange abkneifen. An den Stecker wird jetzt unser Kabel angelötet. Besonders beim 664 ist zu beachten, daß später der Stecker noch richtig angeschlossen werden kann.

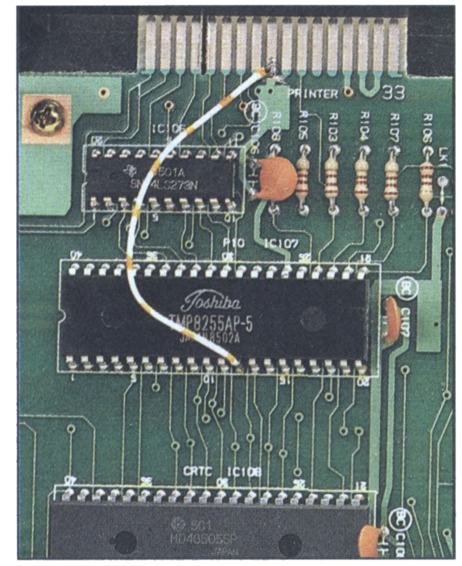


Bild 1. Der Draht beim 664 ist vier Zentimeter lang

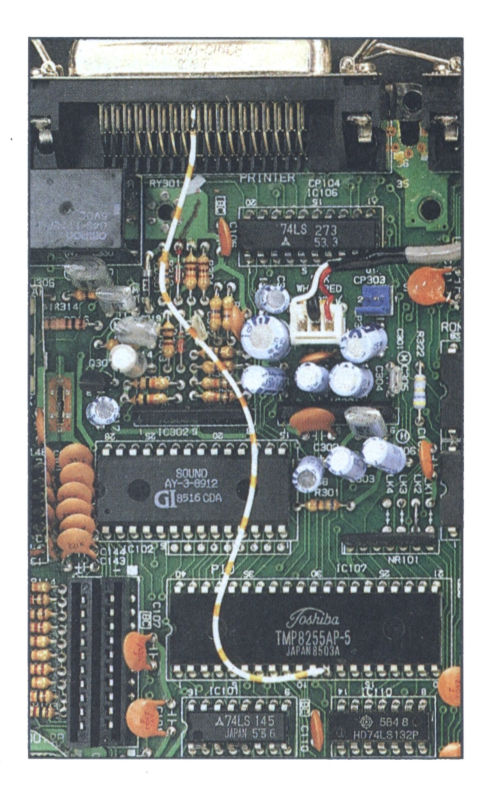


Bild 2. 10 Zentimeter Draht und das achte Bit ist auch beim G 128 da

Man darf also nicht zu weit am Rand der Platine löten. Das zweite Ende der Drahtbrücke kommt an Pin 12 des 8255-Bausteins. Im Bild erkennen Sie deutlich, wo dieser Chip sitzt. Er ist außerdem mit der Zahl 8255 gekennzeichnet. Die Beinchen der ICs sind ab der Einkerbung gegen den Uhrzeigersinn durchnummeriert. Für uns bedeutet das, daß wir die Platine so legen, daß

```
A000           1000          ORG   &A000
A000           1010 BUSY     EQU   &0848; für 664
A000           1010 BUSY     EQU   &0858; für 6128
A000           1020 SEND     EQU   &A00F
A000           1030 WEITER   EQU   &A01C
A000 013200    1040          LD    bc,&32
A003 CD1B08    1050 WARTE    CALL  busy           ;MC BUSY PRINTER
A006 3007      1060          JR    nc,send
A008 10F9      1070          DJNZ  warte
A00A 0D        1080          DEC   c
A00B 20F6      1090          JR    nz,warte
A00D B7        1100          OR    a
A00E C9        1110          RET
A00F           1120                               ;Beginn der 8-Bit-Routine
A00F C5        1130 SEND     PUSH  bc             ;BC Register retten
A010 06F6      1140          LD    b,&F6          ;8255 Port C
A012 CB7F      1150          BIT   7,a            ;Bit 7 im Akku gesetzt?
A014 2806      1160          JR    z,weiter       ;wenn nicht normal weiter
A016 F5        1170          PUSH  af             ;Datenbyte retten
A017 3E20      1180          LD    a,&20          ;Bit 5 setzen und
A019 ED79      1190          OUT   (c),a          ;und an Port C ausgeben
A01B F1        1200          POP   af             ;Datenbyte zurueck
A01C 06EF      1210 WEITER   LD    b,&EF
A01E E67F      1220          AND   &7F            ;Byte ohne Strobe
A020 ED79      1230          OUT   (c),a          ;an Drucker
A022 F680      1240          OR    &80            ;Strobe Bit setzen
A024 F3        1250          DI
A025 ED79      1260          OUT   (c),a          ;Strobe ein
A027 E67F      1270          AND   &7F            ;Strobe zuruecksetzen
A029 FB        1280          EI
A02A ED79      1290          OUT   (c),a          ;Strobe aus
A02C F5        1300          PUSH  af
A02D 06F6      1310          LD    b,&F6          ;Port C
A02F 3E00      1320          OUT   (c),a          ;ausschalten
A031 ED79      1330          LD    a,&00          ;Bit 5 an Port C
A033 F1        1340          POP   af
A034 C1        1350          POP   bc
A035 37        1360          SCF
A036 C9        1370          RET
```

Listing 1. Das kommentierte Assembler-Listing

```
10 MEMORY &9FFF                                        [8A56]
20 FOR z=&A000 TO &A036                                [3034]
30 READ b:POKE z,b                                     [FA64]
40 NEXT                                                [B986]
50 POKE &BDF2,&0                                       [EEF4]
60 POKE &BDF3,&A0                                      [B67A]
70 DATA &01,&32,&00,&cd,&48,&08,&30,&07                [9E60]
80 DATA &10,&f9,&0d,&20,&f6,&b7,&c9,&c5                [E80A]
90 DATA &06,&f6,&cb,&7f,&28,&06,&f5,&3e                [C21C]
100 DATA &20,&ed,&79,&f1,&06,&ef,&e6,&7f
                                                       [76D4]
110 DATA &ed,&79,&f6,&80,&f3,&ed,&79,&e6
                                                       [70F4]
120 DATA &7f,&fb,&ed,&79,&f5,&06,&f6,&3e
                                                       [E248]
130 DATA &00,&ed,&79,&f1,&c1,&37,&c9                   [C294]
```

Listing 2. Der Basic-Lader für den CPC 664

```
70 DATA &01,&32,&00,&cd,&58,&08,&30,&07    [4B62]
```

Listing 3. Für den CPC 6128 muß Zeile 70 geändert werden

der 8255 mit seiner Nase (Einkerbung) nach links vor uns liegt. Nun zählen wir an der unteren Reihe von links nach rechts 12 Beine ab und schon haben wir den gesuchten Port C. Beim Anlöten des Drahtes muß man unbedingt daran denken, daß bei einer Temperatur von 270 Grad maximal 10 Sekunden am IC-Anschluß gelötet werden darf. Also nicht das »arme« IC mit dem Lötkolben »braten«. Jetzt montieren Sie Ihren Schneider wieder zusammen und der Umbau ist fertig.

Über eins müssen Sie sich vor dem Zusammenbau im Klaren sein: Sobald Sie Ihren Computer auseinandernehmen, erlischt Ihr Garantieanspruch für das Gerät. Da der Eingriff aber sehr einfach ist, kann nomalerweise nichts passieren.

Einen ersten Test können wir von Basic aus vornehmen. Mit »OUT &7600,32« wird Bit 5 (Port C am 8255) gesetzt. Eine Schleife »FOR i=128 TO 255:PRINT #8 CHR$(i);:NEXT i« bringt nun den gesamten Grafikzeichensatz des Schneiders zu Papier. Vorausgesetzt, Sie benutzen einen Drucker, der diese Symbole im ROM gespeichert hat. Das sind in aller Regel die Geräte, die den Namenszusatz »CPC«, »für Schneider« oder ähnliches tragen. Andere Drucker geben jeweils ihren eigenen speziellen Zeichensatz aus, der bei ihnen im ROM gespeichert ist. Benutzen Sie ein Gerät, das mit einem frei definierbaren RAM-Zeichensatz arbeitet, so können Sie hier die Symbole des Schneiders speichern und mit unserem Umbau benutzen. Wie Sie die Zeichen in den Speicher des Druckers bekommen, entnehmen Sie dem Handbuch, da dieser Vorgang bei jedem Gerät anders ist.

Mit »OUT &F600,0« setzen Sie das achte Bit wieder zurück. Nun wird der normale ASCII-Code an den Drucker geschickt. Unsere Maschinencode-Routine (Listing 1) bewirkt dieses Umschalten automatisch. Die Routine ist für beide Computer (den 664 und 6128) bis auf den Aufruf der Firmware-Routine MC-BUSY-PRINTER gleich. Listing 2 zeigt den Basic-Lader für den 664 und Listing 3 die Zeile 70, die für den 6128 zu ändern ist. Nach dem Eintippen müssen Sie das Programm einmal mit »RUN« starten, damit es hinter die Adresse A000 hex geschrieben wird. Bis zum Ausschalten (oder bis zu einem Reset mit ESC, SHIFT und CTRL) des Computers bleibt die Betriebssystemveränderung im Speicher stehen. Den Basic-Lader dürfen Sie natürlich löschen.

Wenn ein 464-Besitzer jetzt auf den Geschmack gekommen ist und seinen Computer auch mit einem echten Centronics-Port versehen will, dann muß er auf das erste Schneider-Sonderheft von Happy-Computer (Ausgabe 2/85) zurückgreifen. Dort ist der gleiche Umbau für den Schneider CPC 464 beschrieben.

(Michael Bauer/hg)



Keine Eingabefehler mit »Explora«



Mit einer Prüfsumme wird jede Zeile, die Sie eingeben, überwacht. Fehler im Listing sind damit fast unmöglich. Aber auch Sonder- und Leerzeichen sind besser zu lesen.

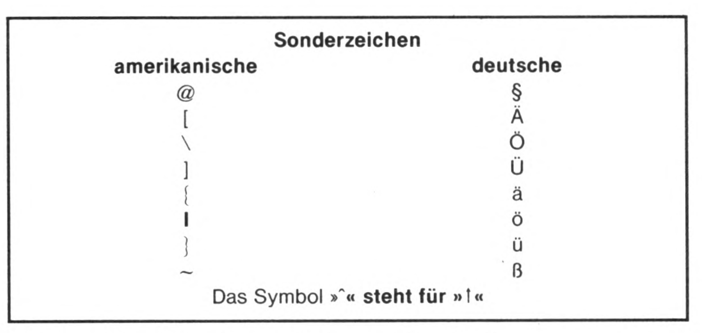
Wenn Sie das Programm »Explora« abtippen, haben Sie eine wertvolle Eingabehilfe. Eine Maschinencode-Routine überwacht Ihre Arbeit daraufhin, ob sämtliche Zeichen (auch Steuersymbole) sowie Leerstellen und Zeilennummer korrekt im Speicher stehen. Nach Beenden einer Zeile mit ENTER, wird direkt in die untere linke Ecke des Bildschirms die vierstellige Hexadezimalzahl angezeigt, die Sie im Listing in der eckigen Klammer neben jeder Programmzeile finden.

Voraussetzung für die Überwachfunktion ist allerdings, daß Sie die Programmzeile genauso eingeben, wie sie abgedruckt ist. Abkürzungen, die vom Interpreter auch verstanden werden, dürfen Sie nicht benutzen (also kein »?« für »PRINT«). Auch müssen Sie große und kleine Buchstaben wie vorgegeben eintippen. Der Interpreter würde für »PRINT« auch »print« akzeptieren — Explora hingegen nicht. Steuerzeichen und mehrere Leerzeichen, die in Strings aufeinander folgen, sind in geschweiften Klammern im Klartext angegeben. So bedeutet {5 Space}, daß an dieser Stelle fünfmal die Leertaste gedrückt werden muß. {CTRL A} bedeutet, daß die Ctrl-Taste gemeinsam mit dem »A« gedrückt werden muß (siehe im Beispiellisting Zeile 430 und 440). Aber Vorsicht, daß Sie solch ein übersetztes Zeichen nicht mit dem ASCII-Sonderzeichen »{« beziehungsweise »}« verwechseln. Die Bedeutung der geschweiften Klammer erkennen Sie aber leicht, denn als ASCII-Sonderzeichen steht sie meist allein. Im anderen Fall umschließt sie Control-Zeichen oder Leerfelder. Der AUTO-Befehl darf übrigens nicht verwendet werden, da sonst die Prüfsumme falsch berechnet wird.

Alle Listings sind mit dem ASCII-Zeichensatz ausgedruckt. Deutsche Sonderzeichen werden dabei als Klammern oder andere amerikanische Sonderzeichen interpretiert. Benutzen Sie eine deutsche Tastatur, so dürfen Sie an Stelle dieser Zeichen die deutschen benutzen. Explora merkt dies. Welche amerikanischen und deutschen Zeichen sich entsprechen, finden Sie in der Tabelle. Das Zeichen » ~ « (für das »ß«) wird mit CTRL 2 aufgerufen.

Listing 1 ist die Routine für die Prüfsumme. Diese liegt ab Adresse 40960 im Speicher. Das Basic-Lader darf gelöscht werden. Eingeschaltet wird Explora mit »POKE & A01F,&F5«, ausgeschaltet mit »POKE &A01F,&C9«. Probleme kann es nur bei Listings geben, die ein Maschinencode-Programm (das sind die Basic-Lader) erzeugen. Eventuell funktioniert der MEMORY-Befehl nicht richtig. In diesem Fall darf er ersatzlos gestrichen werden. Beim Speichern der Binärfelder müssen alle Adressen aber genau beachtet werden.

Der Übersicht halber sind die Zeilen-Nummern nach links herausgezogen. Die eigentlichen Befehle beginnen immer nach einer Leerstelle hinter der Zeilennummer. Listing 2 zeigt fünf Zeilen als Beispiel. Übrigens: Alle Programme können auch, wie gewohnt, ungeprüft eingegeben werden. (hg)

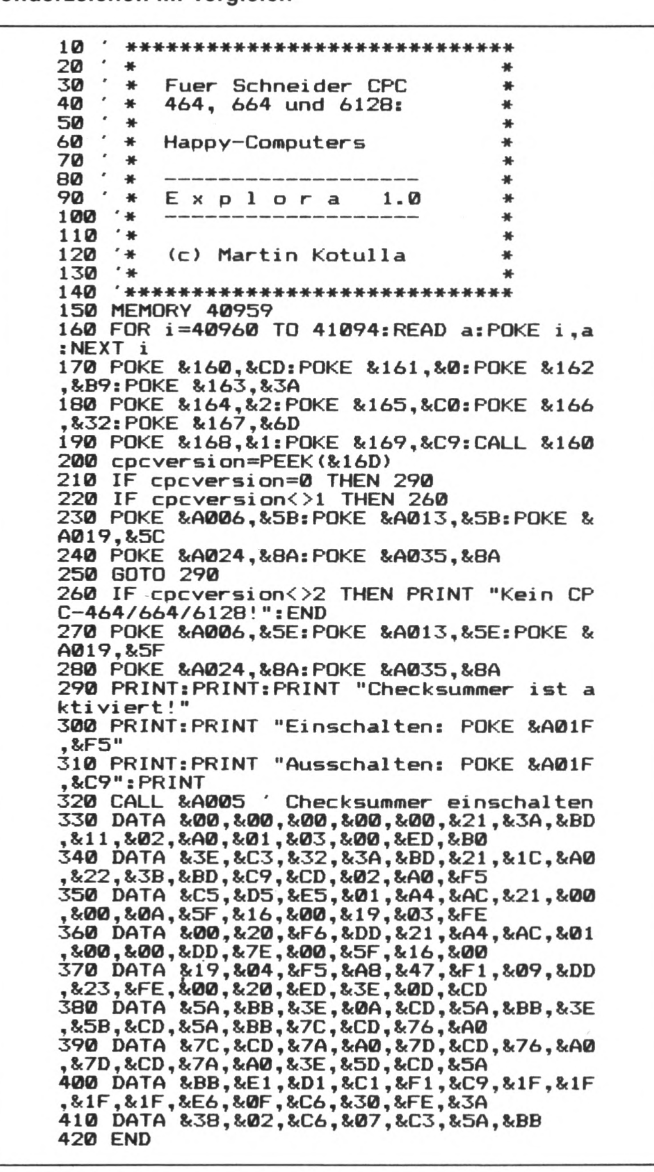


Sonderzeichen

amerikanische	deutsche
@	§
[	Ä
\	Ö
]	Ü
{	ä
|	ö
}	ü
~	ß

Das Symbol »ˆ« **steht für »↑«**

Tabelle. Die deutschen und die amerikanischen Sonderzeichen im Vergleich

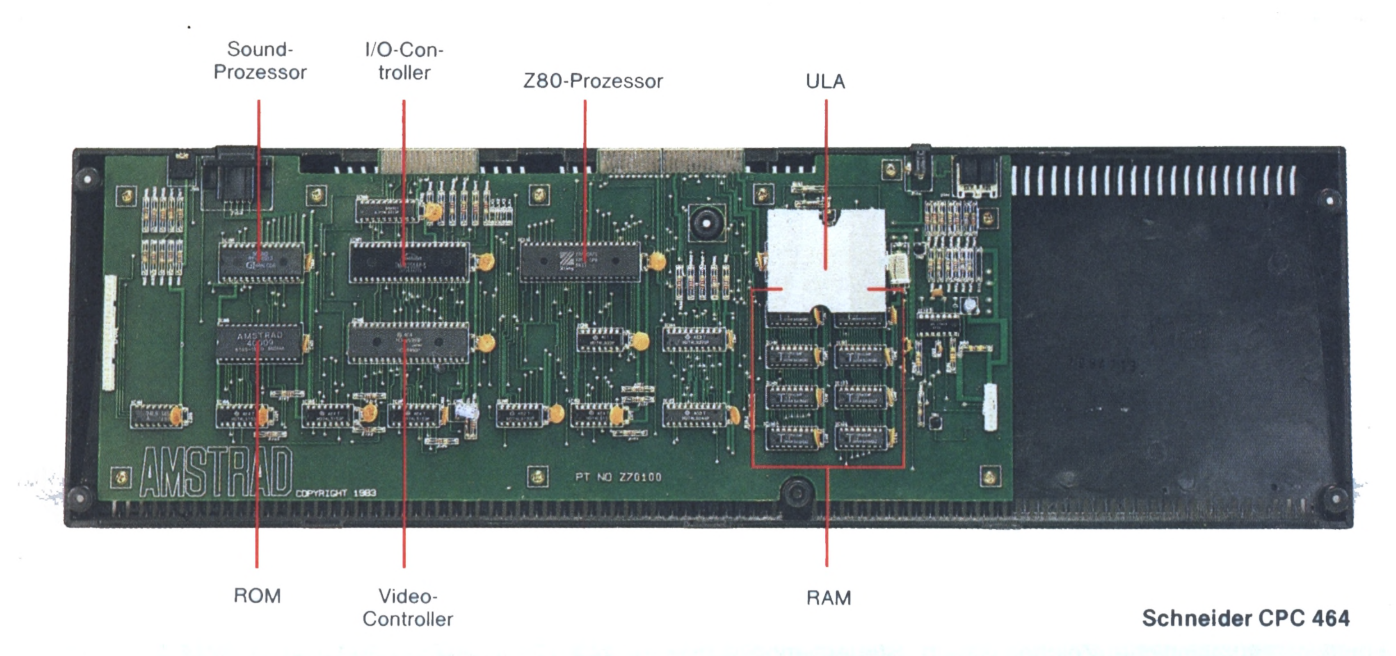


```
10 ' ******************************
20 ' *                            *
30 ' *  Fuer Schneider CPC        *
40 ' *  464, 664 und 6128:        *
50 ' *                            *
60 ' *  Happy-Computers           *
70 ' *                            *
80 ' *  -------------------       *
90 ' *  E x p l o r a   1.0       *
100 '*  -------------------       *
110 '*                            *
120 '*  (c) Martin Kotulla        *
130 '*                            *
140 '******************************
150 MEMORY 40959
160 FOR i=40960 TO 41094:READ a:POKE i,a
:NEXT i
170 POKE &160,&CD:POKE &161,&0:POKE &162
,&B9:POKE &163,&3A
180 POKE &164,&2:POKE &165,&C0:POKE &166
,&32:POKE &167,&6D
190 POKE &168,&1:POKE &169,&C9:CALL &160
200 cpcversion=PEEK(&16D)
210 IF cpcversion=0 THEN 290
220 IF cpcversion<>1 THEN 260
230 POKE &A006,&5B:POKE &A013,&5B:POKE &
A019,&5C
240 POKE &A024,&8A:POKE &A035,&8A
250 GOTO 290
260 IF cpcversion<>2 THEN PRINT "Kein CP
C-464/664/6128!":END
270 POKE &A006,&5E:POKE &A013,&5E:POKE &
A019,&5F
280 POKE &A024,&8A:POKE &A035,&8A
290 PRINT:PRINT:PRINT "Checksummer ist a
ktiviert!"
300 PRINT:PRINT "Einschalten: POKE &A01F
,&F5"
310 PRINT:PRINT "Ausschalten: POKE &A01F
,&C9":PRINT
320 CALL &A005 ' Checksummer einschalten
330 DATA &00,&00,&00,&00,&00,&21,&3A,&BD
,&11,&02,&A0,&01,&03,&00,&ED,&B0
340 DATA &3E,&C3,&32,&3A,&BD,&21,&1C,&A0
,&22,&3B,&BD,&C9,&CD,&02,&A0,&F5
350 DATA &C5,&D5,&E5,&01,&A4,&AC,&21,&00
,&00,&0A,&5F,&16,&00,&19,&03,&FE
360 DATA &00,&20,&F6,&DD,&21,&A4,&AC,&01
,&00,&00,&DD,&7E,&00,&5F,&16,&00
370 DATA &19,&04,&F5,&A8,&47,&F1,&09,&DD
,&23,&FE,&00,&20,&ED,&3E,&0D,&CD
380 DATA &5A,&BB,&3E,&0A,&CD,&5A,&BB,&3E
,&5B,&CD,&5A,&BB,&7C,&CD,&76,&A0
390 DATA &7C,&CD,&7A,&A0,&7D,&CD,&76,&A0
,&7D,&CD,&7A,&A0,&3E,&5D,&CD,&5A
400 DATA &BB,&E1,&D1,&C1,&F1,&C9,&1F,&1F
,&1F,&1F,&E6,&0F,&C6,&30,&FE,&3A
410 DATA &38,&02,&C6,&07,&C3,&5A,&BB
420 END
```

Listing 1. »Explora« macht Fehler fast unmöglich

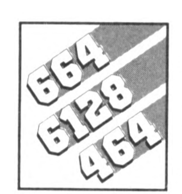
```
400 DATA &BB,&E1,&D1,&C1,&F1,&C9,&1F,&1F
     ,&1F,&1F,&E6,&0F,&C6,&30,&FE,&3A       [C4C0]
410 DATA &38,&02,&C6,&07,&C3,&5A,&BB        [633C]
420 NEW                                     [EE40]
430 PRINT "{CTRL A}{CTRL Y}{CTRL Y}{CTRL
      A}"                                   [8DB8]
440 PRINT "{5 SPACE}WW}{"                   [8CDE]
```

Listing 2. Im Beispiel müßten Sie die Zeile 400 wie folgt eingeben (MODE 1):
400 DATA &BB,&E1,&D1,&C1,&F1,&C9,&1F,&1F ,&1F,&1F,&E6,&0F,&C6,&30,&FE,&3A.
Zeile 430 besteht in der PRINT-Anweisung aus den vier Tastendrucken Ctrl A, Ctrl Y, Ctrl Y und Ctrl A. Zeile 440 aus dem String "   WW}{"



Schneider CPC 464

Aus dem Schneider



Anders als viele ihrer Vorfahren zeichnet die Schneider-Computer ein sehr starkes Basic aus. Aber auch ihr elektronisches Innenleben bietet reichhaltigen Stoff.

Schon bei der ersten, anfänglichen Betrachtung stößt man auf eine der Besonderheiten. Da wird beispielsweise die Hinter- und Vordergrundfarbe nicht direkt über einen Farbcode eingegeben, sondern sie müssen erst mit »INK« Farbregister definiert werden. Es stellt sich natürlich die Frage, warum diese etwas ungewöhnliche Methode? Wir wollen einmal versuchen, diesen Spezialitäten auf den Grund zu gehen. Beginnen wir mit einem groben Überblick. Als Grundlage unserer Betrachtungen dient dabei das Blockschaltbild der Schneider-Computer (Bild 1). Es stellt ein Prinzip-Schaltbild der Computer dar, ist also eine grobere, aber dadurch auch übersichtlichere Variante eines Schaltplans. Beschäftigen wir uns zunächst einmal mit den einzelnen Baustein-Gruppen, aus denen der CPC besteht.

Herzstück des Systems ist ein Z80A-Prozessor, auch als CPU (Central Processing Unit = zentrale Prozeßeinheit) bezeichnet. Dieser Baustein stellt quasi das Gehirn unseres Computers dar. Seine Aufgabe besteht in der Kontrolle und Ausführung des internen und externen Datentransfers und in der Durchführung arithmetischer und logischer Funktionen.

Zur Durchführung dieser Aufgaben benötigt ein Prozessor zunächst einmal Anweisungen, ein Maschinencode-Programm also, die sogenannte Firmware. Dieses Programm muß in einem externen Speicher vorliegen und dort wenigstens teilweise ständig verfügbar, also resident, sein. Daneben benötigt er Speicherplatz zur Ablage von Zwischenergebnissen und Kommunikationsmittel zum Datenaustausch mit seiner Umwelt, Verbindungsglieder zur Tastatur, zum Bildschirm und natürlich auch zu Massenspeicher-Medien (Kassette oder Diskette).

Auch Computer brauchen Kommunikation

Die beim Schneider verwendete Z80A-CPU unterscheidet dabei zwei Arten von peripheren Bausteinen, Memory- und I/O-Bausteine. Bei den Memory-Bausteinen handelt es sich vereinfacht gesagt um Speicher; also um RAM und ROM. Die nicht flüchtigen, permanenten ROM-Bausteine (Read-Only-Memory = Nur-Lese-Speicher) enthalten dabei alle Anweisungen für den Schneider, um die verschiedenen Systemfunktionen auszuführen. Dazu gehört die Ausgabe von Zeichen an den Bildschirm genauso wie das Laden von Programmen und andere grundlegende Operationen. Dazu kommt noch eine Vielzahl von Konstanten, wie beispielsweise die Ursprungs-Definition aller Zeichen, die Anfangsbelegung der Tastatur aber auch der auf zehn Stellen genaue Wert der Zahl »PI«. In den RAM-Bausteinen werden veränderliche Informationen abgelegt (»Random-Access-Memory« bedeutet Schreib-/Lese-Speicher mit beliebigem Zugriff«), wie beispielsweise der Bildschirminhalt, ein eingegebenes Basic- oder CP/M-Programm.

Alle anderen Bausteine, wie I/O-Chips, die die Kommunikation mit der Außenwelt sicherstellen oder die Video-Bausteine zur Ausgabe von Bildschirminformation über den Monitor, werden als I/O angesprochen. Das Kürzel »I/O« steht dabei für Input/Output, also Ein-/Ausgabe-Baustein.

Wenn Sie das Blockschaltbild betrachten, können Sie eine grobe Dreiteilung des Systems vornehmen. Ganz unten finden Sie die CPU. Darüber liegt die I/O-Gruppe und im oberen Teil des Bildes dann der Memory-Bereich. Hier gibt es allerdings eine Einschränkung: Das ganz oben liegende ULA (mehr zu seinen Aufgaben im zweiten Teil dieses Artikels) wird als I/O-Gerät angesprochen.

Alle Bausteine sind durch einen großen Leitungsstrang, den sogenannten Bus, verbunden, der sich in drei Teile gliedert:

- Der Adreßbus:

Er besteht aus 16 Leitungen (=16 Bit) und dient zur Adressierung, das heißt Auswahl eines bestimmten I/O-Bausteins, beziehungsweise einer Adresse im Speicher. Hierbei handelt es sich um reine Ausgabeleitungen. Die CPU setzt eine Adresse und erwartet dann von dieser Speicherstelle Daten, beziehungsweise sendet an diese Informationen. Die 16 Adreßleitungen

haben Nummern von 0 bis 15. A0 (Adreßleitung Nummer 0) stellt dabei das niederstwertige Bit dar, A15 das höchstwertige. Mit 16 Leitungen lassen sich 2^{16} verschiedene Kombinationen darstellen, oder in diesem Fall 2^{16} =65536 verschiedene Speicherstellen adressieren.

- Der Datenbus:

Er besteht aus acht Leitungen (=8 Bit) und kann wahlweise auf Eingabe, das heißt den Empfang von Daten oder Ausgabe programmiert werden. Dies geschieht jeweils durch den Prozessor bei der Ausführung eines Maschinenbefehls. Beim Z80A werden immer 8 Bit (= 1 Byte) parallel gesendet beziehungsweise empfangen. Die Datenbus-Leitungen tragen Nummern von 0 bis 7. Die 0 stellt wieder das niederstwertige, die 7 das höchstwertige Bit dar.

- Der Steuerbus:

Beim Steuerbus handelt es sich um einige weitere Leitungen, die steuernde Funktionen wahrnehmen. Wir haben bei der Beschreibung des Adreßbusses schon errechnet, daß die CPU maximal 64 KByte (= 65536) verschiedene Speicherstellen direkt adressieren kann. Dies reicht normalerweise nicht aus. Denn schon das RAM, der variable Speicher unseres Computers, belegt 64 KByte; beim CPC 6128 sind es sogar 128 KByte. Deshalb hat man bei der Entwicklung des Z80A-Prozessors die schon beschriebene Trennung in I/O- und Memory-Bausteine eingeführt. Durch zwei Leitungen im Steuerbus wird angezeigt, ob die CPU einen Ein-/Ausgabe-Baustein oder ein Speicher-IC ansprechen will. Dadurch steht die doppelte Anzahl von Speicherstellen zur Verfügung. Jeder Baustein im System ist nämlich mit einer der beiden Leitungen verbunden und nur, wenn neben dem Adreßsignal auch noch diese Freigabe-Leitung den richtigen Wert aufweist, schaltet er sich in die Kommunikation ein. Nun ist Kommunikation eine schwierige Sache. Irgend jemand muß beispielsweise einem Speicherbaustein ja auch noch sagen, ob dieser »zuhören«, das heißt Daten empfangen oder »sprechen«, also Daten senden soll. Dazu existieren zwei weitere Leitungen im Bus: eine, die ein Schreibsignal (aus Sicht des Prozessors!) ausgibt, eine andere, die Informationen anfordert. Daneben gibt es im Steuerbus noch weitere Verbindungen zur Abstimmung des Datenaustausches zwischen den einzelnen Bausteinen, also zur Synchronisation.

Daten-, Adreß- und Steuerbus verbinden als Leitungsstrang alle wesentlichen Baugruppen des CPC. In unserem Schaubild symbolisiert ein Balken in der Bildmitte den Bus.

Wenn im Schneider CPC nun beispielsweise ein Wert an einen I/O-Baustein ausgegeben werden soll, läuft folgendes ab:

Der Z80 gibt auf dem Adreßbus den Code des anzusprechenden Ausgabebausteins aus. Gleichzeitig wird auf dem Steuerbus angezeigt, daß ein Ein-/Ausgabe-Baustein aktiv werden und Daten empfangen soll. Die Daten werden fast gleichzeitig auf dem Daten-Bus zur Verfügung gestellt.

Betrachten wir nun einmal, was die einzelnen Bausteine im Zusammenwirken alles leisten.

Centronics-Port als Eingabe-Schnittstelle

Die Z80-CPU als Schaltzentrale der Schneider-Computer haben wir mit ihren verschiedenen Funktionen schon kennengelernt. Als nächstes wollen wir uns daher mit den im Blockschaltbild darüber liegenden Bausteinen beschäftigen, den Schaltkreisen des I/O-Bereichs.

Da wäre zunächst einmal der Druckerausgang, auch als Centronics-Port bezeichnet. Dieser Ausgang stellt die Schnittstelle zum Drucker dar, kann jedoch auch als allgemeiner Steuerausgang benutzt werden. Die Ausgabe von Informationen an den Drucker erfolgt nicht direkt, sondern unter Zwischenschaltung eines 8-Bit-Schieberegisters. Dieses Register ist als 8-Bit-Ausgaberegister geschaltet. Es übernimmt auf Kommando, bei Anlegen eines Taktsignals, Daten vom Datenbus und speichert diese bis zum Auftreten des nächsten Taktes zwischen. Leider werden schaltungsbedingt nur 7 Bit ausgegeben. Wie diesem Problem beizukommen ist, darüber wurde bereits im letzten Sonderheft (Ausgabe 1/86) ausführlich berichtet. Hier wollen wir daher auf einen anderen Bereich genauer eingehen: Die Benutzung des Druckerausgangs als universellem Ausgabekanal.

Der Centronicsport stellt eine sehr einfach zu bedienende, universelle Aus- und, was noch meist unbekannt ist, Eingabeschnittstelle dar. Neben den sieben Datenleitungen und dem Strobe-Signal steht am Printerport nämlich auch noch ein 1-Bit-Eingabekanal

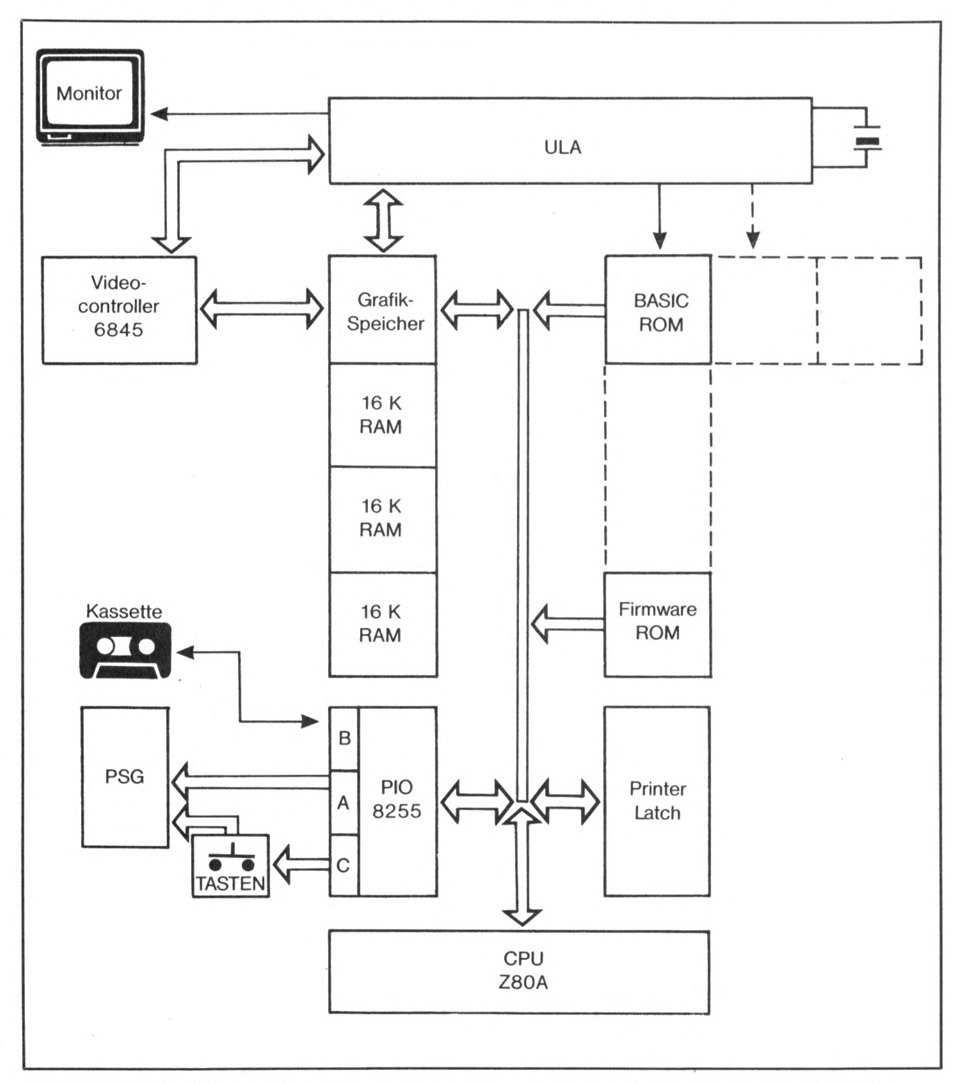
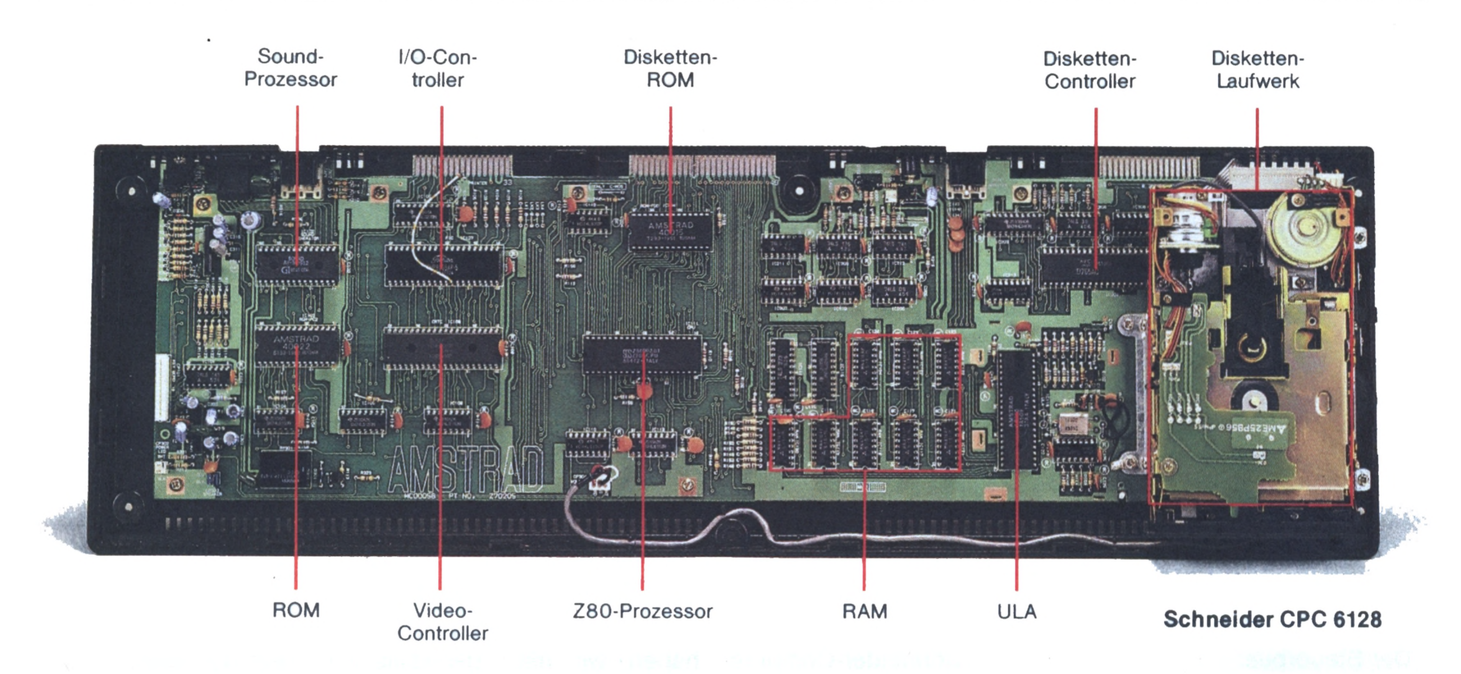


Bild 1. So sind die einzelnen Bausteine verbunden. Das abstrakte Blockschaltbild zeigt einen groben Überblick über die Verbindung der einzelnen Bausteine.



Schneider CPC 6128

zur Verfügung, der Busy-Eingang. Nomalerweise wird dieser Eingang vom Drucker benutzt, um anzugeben, daß er mehr Zeichen empfangen hat, als er aktuell auswerten kann, und der Computer sich doch bitte etwas mit der weiteren Ausgabe gedulden möge. Man kann diesen Kanal jedoch auch extern beschalten und dann für universelle Eingabe verwenden.

Um den Centronicsport zu benutzen, muß man sich zunächst einmal überlegen, wo die verschiedenen Signale zur Verfügung stehen. Dazu genügt ein kurzer Blick in den Anhang Ihres Benutzerhandbuches, in dem auch die Belegung des Druckerausgangs beschrieben ist.

Wenn Sie sich den Schneider von hinten ansehen, liegt Pin 1 dabei auf der Oberseite der Platine, beziehungsweise beim CPC 6128 ganz rechts außen. Von rechts nach links nehmen dann die Pin-Nummern zu. Pin 1 liefert dabei das Strobe-Signal. Die nächsten sieben Pins stellen die Datenleitungen dar. Pin 9, die achte Datenleitung, ist beim Schneider permanent auf 0 gelegt. Pin 10 wird nicht genutzt und Pin 11 ist mit dem Busy-Eingang verbunden.

Nachdem wir nun wissen, wie die Ausgänge beschaltet sind, können wir uns dem eigentlichen Datentransfer zuwenden. Die Nutzung des Centronicsports ist in Basic mit dem OUT-Befehl problemlos möglich.

»OUT &EFFF,<Wert>« gibt den »Wert« an den Druckerausgang aus. Dieser bleibt dort solange erhalten, bis wir ihn überschreiben. Wenn Sie also beispielsweise »OUT &EFFF,&x1100 1100« eingeben und dann mit einem Meßinstrument Pin 4 testen (Masse des Instruments muß dabei auf der Unterseite der Platine liegen), lesen Sie hier +4 Volt ab, also eine logische 1.

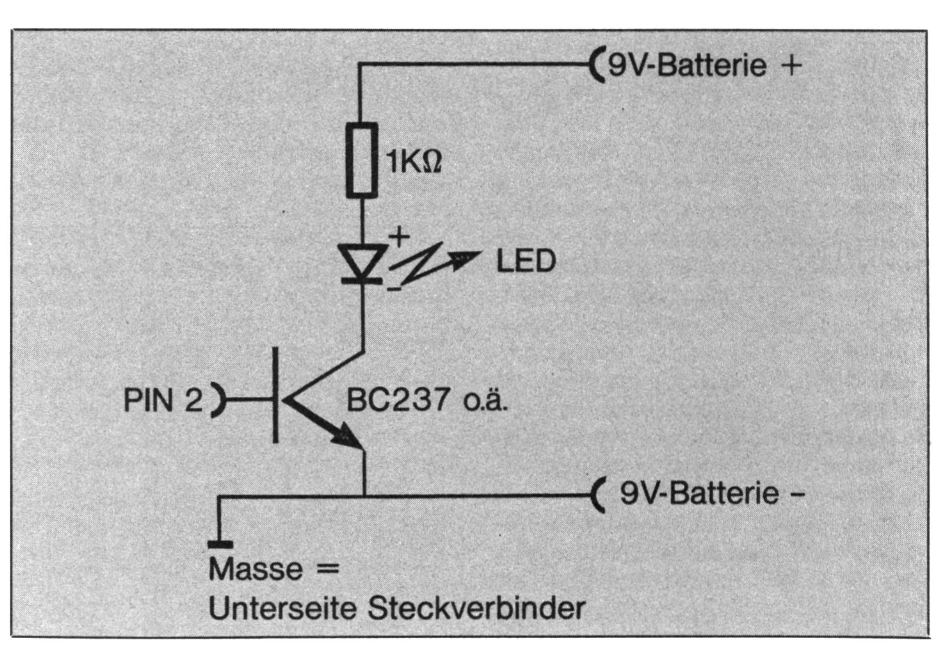


Bild 2. Der CPC läßt eine Leuchtdiode munter blinken

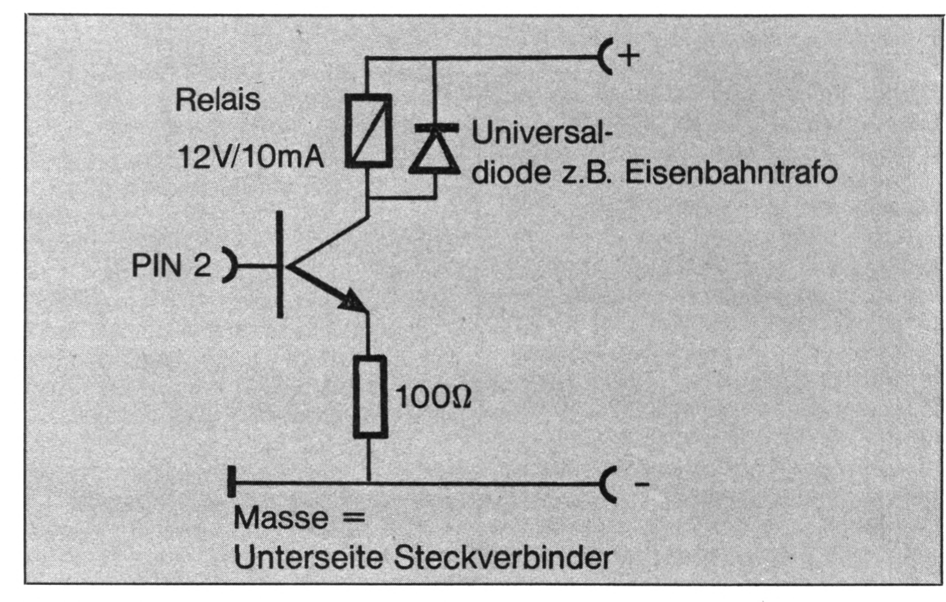


Bild 3. Eine universelle Steuerung am Druckerport mit wenigen Bauteilen

Professionelle Grafikprogramme für Schneider CPC 6128 + Joyce



DR Draw

DR Draw: Macht aus Ihren Ideen ein Kunstwerk

Verwenden Sie DR Draw, um Organisations-Diagramme, Flußdiagramme, Logos, technische Zeichnungen, Schaubilder, Platinenentwürfe und jede nur erdenkliche Art von Linien- und Formgrafiken zu entwerfen. Und jeder Bestandteil Ihrer Zeichnung kann auf vielfältige Weise durch Farben und Schraffuren hervorgehoben werden.

Einfachste Bedienung

DR Draw verwendet leichtverständliche Menüs zur Steuerung seiner Funktionen und Erstellung einer Zeichnung. Sie können aus vorprogrammierten Figuren wie Kreisen, Quadern, Rechtecken, Kreisbögen, Polygonen und Linien auswählen oder Ihre eigenen Figuren entwerfen oder die bestehenden verändern. An beliebigen Stellen kann erläuternder Text in eine Zeichnung eingefügt werden. Außerdem haben Sie die Wahl zwischen mehreren Schriftarten.

Flexibilität bei der Gestaltung

Jeder Teil einer Zeichnung kann auf Tastendruck überarbeitet und verändert werden: Figuren können mit Farben oder Mustern gefüllt werden; sie können vergrößert oder verkleinert oder an eine neue Position verschoben oder kopiert werden. Ebenso können die Schriftarten, Größen, Farben und Positionen mit wenigen Tastendrücken geändert werden.

Vergrößerungen und Ausschnittdarstellungen

Mit Hilfe einer besonderen Funktion von DR Draw können Sie Einzelheiten Ihrer Zeichnung vergrößern, um Details besser bearbeiten zu können.

Ausgabe auf Papier, Transparentfolie oder Film

Was immer Sie erstellen, kann gespeichert oder zu Berichts- und Präsentationszwecken auf Papier, Transparentfolie oder Film geplottet oder gedruckt werden. DR Draw druckt Ihre Zeichnung exakt auf eine DIN-A4-Seite.

Hardwarevoraussetzungen

DR Draw läuft auf jedem Schneider CPC 6128 oder Joyce PCW 8256 mit einem oder zwei Diskettenlaufwerken. Die Grafiken können auf jedem Drucker oder Plotter ausgegeben werden, für den ein GSX-Treiber verfügbar ist. Dazu zählen Schneider-, Epson- und Shinwa-Drucker sowie der Plotter HP 7470A.

Die Fähigkeiten auf einen Blick

- Erstellung beliebiger Zeichnungen
- vorprogrammierte Figuren wie Kreise, Quader, Rechtecke, Kreisbögen, Polygone und Linien
- freie Wahl der Gestaltungselemente wie Farben, Muster und Schriftarten
- Vergrößerungen und Ausschnittdarstellungen
- Teile einer Zeichnung können kopiert, verschoben oder gelöscht werden
- Grafiken können gespeichert, geplottet oder gedruckt werden
- einfache Bedienung durch Menüauswahl

Best.-Nr. MS 613 **DM 199,–*** (sFr. 178,–/öS 1890,–*)

DR Graph

DR Graph: Präsentationsgrafiken mit professionellem Niveau

DR Graph ist ein interaktives Softwarepaket, mit dem Sie Ihren Mikrocomputer zur Erstellung von Geschäftsgrafiken und Text-Charts verwenden können. DR Graph macht es leicht, komplexe geschäftliche oder wissenschaftliche Daten in übersichtliche und aussagekräftige Grafiken zu verwandeln.

Ein Bild sagt mehr als tausend Worte

Eine gut dargestellte Grafik weckt das Interesse und die Aufmerksamkeit des angezielten Personenkreises eher als andere Kommunikationsarten. Grafisch dargestellte Fakten können leichter analysiert, verstanden und behalten werden.

Einfachste Bedienung

Mit DR Graph können Sie die Grafik dem Computer schnell und leicht beschreiben. Zur Erstellung einer Grafik werden die gewünschten Optionen ganz einfach aus übersichtlichen Menüs ausgewählt. DR Graph kann von jedermann bedient werden, der mit einfachen Grundlagen der Mikrocomputerbedienung vertraut ist.

Flexibilität bei der Gestaltung

Zusätzlich zur vorhandenen Computerschrift stehen drei verschiedene Schriften für Titelzeilen, Legenden und Anmerkungen zur Verfügung. Auch bei der Gestaltung der Grafiken kann aus zahlreichen Linientypen, Linien- und Balkenbreiten und acht Schraffuren gewählt werden.

Ansehen, speichern und drucken

Mit DR Graph können Sie auf dem Bildschirm immer genau sehen, wie Sie Ihre Grafik gestalten. Anschließend können Sie sie drucken oder auf Diskette speichern, um sie später weiter zu bearbeiten.

Hardwarevoraussetzungen

DR Graph läuft auf jedem Schneider CPC 6128 oder Joyce PCW 8256 mit einem oder zwei Diskettenlaufwerken. Die Grafiken können auf jedem Drucker oder Plotter ausgegeben werden, für den ein GSX-Treiber verfügbar ist. Dazu zählen Schneider-, Epson- und Shinwa-Drucker sowie der Plotter HP 7470A.

Die Fähigkeiten auf einen Blick

- Linien-Grafiken, Histogramme, Torten-Grafiken, Stufen-Grafiken, Strich-Histogramme, Punkte-Grafiken und Text-Grafiken
- freie Wahl der Gestaltungselemente wie Beschriftungen, Titelzeilen, Legenden, Farben, Schriftarten und Ränder
- frei wählbare Skalierung
- variable Linien- und Balkenbreite
- Schnittstelle zu anderen Programmen
- beliebig positionierbare Anmerkungen
- Grafiken können gespeichert, geplottet oder gedruckt werden
- einfache Bedienung durch Menüauswahl

Best.-Nr. MS 614 **DM 199,–*** (sFr. 178,–/öS 1890,–*)

In Vorbereitung:

Fakturierung

Ein dBASE-II-Anwenderprogramm, das folgende Möglichkeiten bietet: Angebotsschreibung und Rechnungsschreibung, Artikelverwaltung, Adreßverwaltung, Nachkalkulation. Der dokumentierte Quellcode wird für individuelle Programmanpassungen mitgeliefert.

Best.-Nr. MS 616 **DM 94,–*** (sFr. 82,–/öS 940,–*)

Finanz-Buchhaltung

Das Komplett-Paket für den Schneider CPC 6128 und Joyce. Erstellen von Kontenplänen, Umsatzsteuerauswertung und Einnahmen-/Überschußrechnung. Betriebswirtschaftliche Auswertungen wie Journalschreibung und Kostenstellenrechnung möglich.

Best.-Nr. MS 615 **DM 94,–*** (sFr. 82,–/öS. 940,–*)

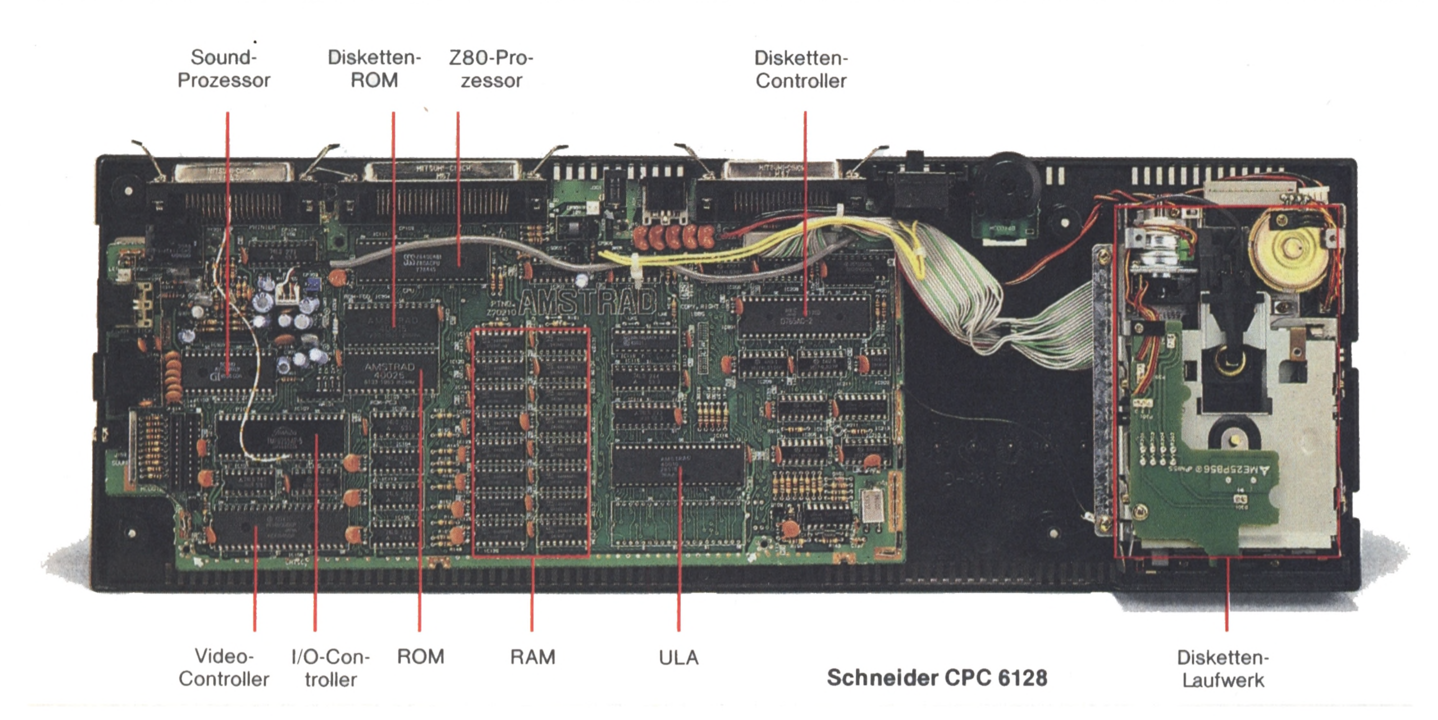
* inkl. MwSt. Unverbindliche Preisempfehlung



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Diese Markt & Technik-Softwareprodukte erhalten Sie in den Fachabteilungen der Kaufhäuser und in Computershops.

Bestellungen im Ausland bitte an untenstehende Adressen.
Schweiz: Markt & Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042/41 56 56
Österreich: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Straße 24, A-1091 Wien, 02 22/48 15 38-0



Schneider CPC 6128

Ebenso an dem danebenliegenden Pin 5, während die ersten beiden Datenleitungen (Pin 2 und Pin 3) keine Spannung aufweisen. Wenn Sie nun »OUT &EFFF,&x00110011« ausgeben, so werden die untersten beiden Pins gesetzt und die nächsthöheren zwei gelöscht. Für die vier oberen Bit gilt natürlich dasselbe. Pin 9, also die Datenleitung D8, können Sie mit dieser Methode nicht verändern, da diese, wie schon weiter oben gesagt, permanent auf 0 gesetzt ist.

Universelle Steuerungen

Sie sollten natürlich bei diesen Experimenten strengstens darauf achten, keinen Kurzschluß zu verursachen. Eine unabsichtliche Verbindung zwischen einem Pin, der eine »1« führt und einem, der auf Masse gelegt ist, kann Ihren Computer zerstören!

In Bild 2 und 3 finden Sie zwei einfache Schaltungen, mit denen Sie den Schneider schnell als Steuercomputer für eine Eisenbahnanlage oder andere Anwendungen verwenden können. Wenn Sie nur mit dieser Möglichkeit experimentieren wollen, genügt die kleine Schaltung in Bild 2, deren Bauteile für nur wenige Mark zu haben sind. Hier steuert Ihr Schneider eine Leuchtdiode. Ist Ihnen dann das Prinzip klar, können Sie mit der anderen Schaltung arbeiten. Dort schaltet der Schneider ein kleines Relais, mit dessen Hilfe Sie problemlos Stromverbraucher kontrollieren können. Hierzu noch ein Hinweis: Erstens sollten Sie niemals direkt am Computer, also beispielsweise ein Kabel auf eine Leiterbahn der Platine, löten. Mit Sicherheit ist in Ihrer Nähe ein Elektronik-Fachgeschäft, wo Sie für diese Experimente einen Stecker für den Druckerausgang kaufen können! Auch sollten Sie von solchen Experimenten ablassen, wenn Ihnen nicht alles klar ist. Im Zweifelsfall ziehen Sie lieber einen befreundeten Hobbyelektroniker zu Rate, als Ihren Computer zu ruinieren.

Nehmen wir einmal an, Sie wollen eine einfache serielle Schnittstelle auf Ihrem Computer realisieren. Auch dies ist von der Hardwareseite her mit dem Druckerausgang möglich. Eine serielle Schnittstelle benötigt nämlich im Prinzip nur zwei Leitungen, eine Eingabe-Leitung und eine Ausgabe-Leitung. Daneben muß bei einer solchen Schnittstelle auch noch eine Masse-Leitung vorhanden sein, aber die haben wir ja sowieso auf der Unterseite unserer Platine reichlich zur Verfügung. Die Ausgabeseite bereitet keine Probleme. Wir benutzen hier einfach eine Datenleitung, wie schon bei den Steueranwendungen. Die Eingabe soll mit Hilfe des Busy-Eingangs vonstatten gehen. Die Ansteuerung der Ausgabeseite ist ebenfalls nicht schwierig. Sie setzen beispielsweise die unterste (D0) Datenleitung auf 1, wenn eine 1 ausgegeben werden soll, beziehungsweise Sie löschen sie, wenn das Gegenteil der Fall ist. In Basic sind die zugehörigen Befehle:

»OUT &EFFF,1« bei Ausgabe einer 1 und

»OUT &EFFF,0« bei Ausgabe einer 0

Wie sieht es nun mit der Eingabe aus? Das Eingabesignal müssen wir an Pin 11 der Steckerleiste anlegen. Die Abfrage gestaltet sich hier allerdings etwas schwieriger, als bei der Ausgabe, weil das Busy-Signal nur über einige Umwege greifbar ist. Es wird nämlich mit Hilfe des universellen I/O-Bausteins 8255 (links in unserem Blockschaltbild) aufgenommen. In Basic sind dazu mehrere Befehle erforderlich, auf die wir noch bei der Beschreibung dieses Bausteins zurückkommen. Es gibt jedoch auch noch die Alternative, zur Auswertung mit Maschinensprache-Routinen zu arbeiten. Für eine schnelle Abfrage oder Sendung von Daten ist dies sowieso unumgänglich. Im Firmware-Bereich der Schneider-CPCs existieren zwei Routinen, die sich für unsere Zwecke bestens eignen:

»MC-BUSY-PRINTER« (RAM-Adresse BD2E hex) und

»MC-PRINT-CHAR« (BD2B hex).

Die erste Routine überprüft, ob ein Signal am Busy-Eingang anliegt. Ist dies der Fall, wird das Carry-Flag auf 1 gesetzt, ansonsten gelöscht. Durch regelmäßigen Aufruf dieser Routine und Auswertung des Carry-Flags können wir also zu jedem Zeitpunkt den Busy-Eingang überprüfen und damit Daten von unserer seriellen Schnittstelle übernehmen. Wenn wir dagegen Daten senden wollen, benutzen wir die zweite Routine. Dazu brauchen wir nur den auszugebenden Wert in den Akkumulator zu laden. Alles weitere erledigt die Routine.

Datentransfer mit Maschinencode

Mit einem kleinen Hilfsprogramm in Maschinensprache können wir allerdings MC-BUSY-PRINTER auch von Basic aus benutzen:

```
CALL MC-BUSY-PRINTER      CD 2E BD
JR NC,$+4                    38 04
LD (42020),A              32 24 A4
RET                             C9
```

```
LD A,1                    3E 01
LD (42020),A           32 24 A4
RET                          C9
```

Sie können dieses Programm an jeder Stelle im RAM ablegen, beispielsweise ab Adresse 42000. Dazu dient der nachfolgende Basic-Lader.

```
10 MEMORY 41999
20 FOR i=42000 TO 42014:READ a$:
POKE i,VAL("&"+a$):NEXT
30 DATA CD,2E,BD,38,04,32,24,A4,
C9,3E,01,32,24,A4,C9
```

Wenn Sie nun den Busy-Eingang abfragen wollen, geben Sie einfach »CALL 42000:PRINT PEEK(42020)« oder »CALL 42000:var=PEEK (42020)« ein. Sie erhalten dann auf dem Schirm, beziehungsweise in der Variablen »var«, den aktuellen Zustand des Busy-Eingangs. Mit einer wiederholten Abfrage können Sie eine laufende überwachung realisieren und die am Eingang anliegenden Signale einlesen.

Auch im »normalen Hausgebrauch« läßt sich das kleine Maschinenprogramm gut verwenden, nämlich zur Abfrage der Druckerbereitschaft. Wenn Sie mit »PRINT # 8« einen String an den Drucker ausgeben und dieser nicht empfangsbereit (zum Beispiel nicht eingeschaltet) ist, wartet der CPC gegebenenfalls ewig auf ein Freigabe-Signal vom Drucker und stoppt bis dahin sämtliche Aktivitäten. Rufen Sie aus Ihrem Basic-Programm aber vor dem Drucken obige Routine mit »CALL 42000« auf und testen dann, ob »PEEK(42020)« =1 ist, können Sie das verhindern.

Daneben findet das Hilfsprogramm allerdings auch noch in einem anderen Zusammenhang Anwendung: bei der parallelen Druckausgabe. Wir haben schon gesagt, daß der CPC solange wartet, bis der Drucker empfangsbereit ist. Während er Daten an den Druckerausgang sendet, kann er nichts anderes tun. Dies ist besonders dann störend, wenn eine große Menge von Daten gleichzeitig auf dem Schirm und dem Drucker ausgegeben werden soll. Da die Drucker-Ausgabe meist relativ langsam vor sich geht - der CPC liefert die Daten bis zu zehnmal schneller, als der Drucker diese aufs Papier bringen kann - könnte der Computer eigentlich neben der Druckeransteuerung noch viele andere Funktionen wahrnehmen.

Die meisten Drucker verfügen über einen relativ großen Zwischenspeicher, der alle ankommenden Daten erst einmal puffert. Es ist daher problemlos möglich, den Pufferspeicher vollzuladen - wenn dieser voll ist, meldet der Drucker »Busy« -und dann den Drucker arbeiten zu lassen, während der CPC sich mit der Bildschirmausgabe beschäftigt. Ist der Druckerspeicher dann wieder leer, was sich zeitlich abschätzen läßt (je nach Drucker dauert es zwischen 3 und 20 Sekunden), kann die Druckausgabe fortgesetzt werden. Unsere Hilfsroutine macht es möglich: Sie schicken einfach solange Daten an den Drucker, bis dieser »Busy« zurückgibt. Dann geht es in einen anderen Programmteil, aus dem Sie nach einer Zeitverzögerung wieder in die Druckausgabe zurückkehren. Die Rückkehr können Sie dabei mit dem EVERY-Befehl veranlassen. Als Beispiel nehmen wir ein Textverarbeitungsprogramm. Sie wollen einen Text, der aus 500 Strings besteht und zeilenweise in dem Variablenfeld z$(1) bis z$(500) gespeichert ist, ausgeben. Gleichzeitig beabsichtigen Sie aber, zum Beispiel mit einer Editier-Routine, in diesem Programm an einem anderen Text Änderungen vornehmen. Dazu könnten Sie eine Programmkonstruktion benutzen, die folgendermaßen aussieht:

Geschwindigkeit durch einen Druckerpuffer

```
10 EVERY 200,2 GOSUB 10000
20 j=0
30 REM restliches Programm
40 .....

10000 'Unterprogramm
10010 j=j+1
10020 IF j>500 THEN REMAIN
      (2):RETURN
10030 PRINT#8,z$(j)
10040 CALL 42000:IF PEEK(42020)
      =1 THEN RETURN
10050 GOTO 10010
```

Durch den EVERY-Befehl erfolgt alle vier Sekunden ein Sprung in das Unterprogramm, solange, bis der Zeitzähler dort durch den REMAIN-Befehl der Druckroutine zurückgestellt wird. Das Druckprogramm selbst gibt in Abhängigkeit von »j« die einzelnen Zeilen nacheinander aus. Nach jeder Zeile ist ein Aufruf der Maschinen-Routine in Adresse 42000 vorgesehen, der überprüft, ob der Speicher des Druckers voll ist. Ist dies der Fall, kehrt die Druckroutine ins Hauptprogramm zurück und sendet dem Drucker für eine Zeitlang (bis zum nächsten EVERY-Befehl) keine neuen Daten.

Nachdem wir uns relativ ausführlich mit dem Druckerport beschäftigt haben, kommen wir nun zum Hauptbaustein im I/O-Bereich: Dem 8255, einem sehr vielseitigen IC, das in vielen Z80-Systemen Verwendung findet. Der 8255 ist von seiner Hauptfunktion her ein Schnittstellenbaustein. Er spielt die Rolle eines »Verkehrspolizisten«, legt also fest, welcher Ein-/Ausgabeteil (Tastatur, Kassettenrecorder etc.) mit der CPU verbunden wird und gibt dabei auch die Datenflußrichtung an. Auf der einen Seite ist der 8255 direkt mit der CPU beziehungsweise dem Datenbus verbunden. Auf der anderen Seite stehen 24 I/O-Leitungen für die Kommunikation mit der Außenwelt zur Verfügung. Diese sind in drei Registern mit den Bezeichnungen A, B und C zusammengefaßt. Jedes der drei Register kann wahlweise auf Ein- oder Ausgabe programmiert werden. Daneben können

Adresse	Ausgabe (OUT)	Eingabe (INP)
F4xx	Port A Daten ausgeben	Port A Daten einlesen
F5xx	Port B Daten ausgeben	Port B Daten einlesen
F6xx	Port C Daten ausgeben	Port C Daten einlesen
F7xx	Steuerport schreiben	

Tabelle. Adressen des Steuerregisters im »8255«

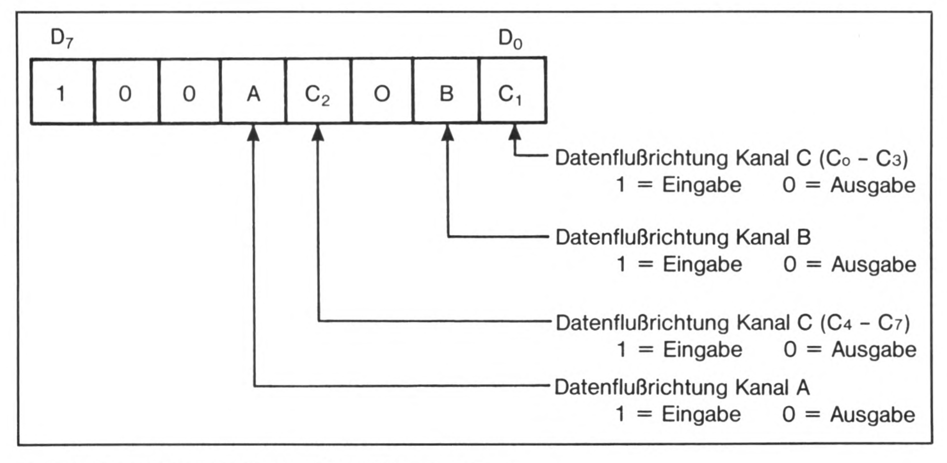


Bild 4. Zusammensetzung eines Steuerwortes

die Register zu Gruppen zusammengefaßt werden, um Steuerleitungen zur Übertragung von Koordinations-Impulsen bei der Datenübertragung, die sogenannten Hand-Shaking-Signale, zu definieren. Die Schneider-Computer machen allerdings von dieser Möglichkeit keinen Gebrauch, so daß wir hier auf eine nähere Beschreibung verzichten.

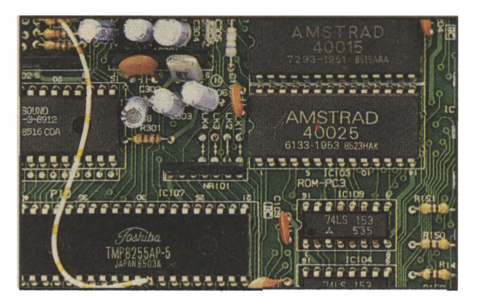
Es wurde bereits erwähnt, daß der 8255 drei Kanäle öffnen kann, die jeweils wahlweise als Input oder Output definiert werden können. Es stellt sich die Frage, wie all diese Betriebsarten und Ein-/Ausgabedefinitionen dem Chip mitgeteilt werden. Dazu wurde bei der Entwicklung im 8255 ein sogenanntes Steuerregister integriert. Durch Senden eines Datenwortes in das Steuerregister wird festgelegt, in welcher Betriebsart der 8255 arbeiten soll und in welcher Richtung die einzelnen Ports dabei wirken sollen. Das Format für ein Steuerwort ist in Bild 4 dargestellt.

»Verkehrspolizist« im Datenstrom

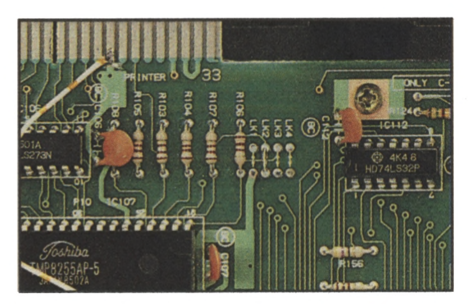
Der 8255 verfügt über zwei Adreßleitungen, die angeben, ob der Datenbus mit dem Port A, B, C oder dem Steuerregister verbunden werden soll. Damit ergeben sich die Adressen für die Ansteuerung der Ports beziehungsweise des Steuerregisters, die in der Tabelle zu sehen sind.

Bevor wir jedoch auf einen Port des 8255 zurückgreifen können, muß erst die Ausgaberichtung im Steuerregister festgelegt sein. Nehmen wir an, wir wollen die Ports A und B als Eingabekanäle betreiben, C jedoch in der umgekehrten Richtung. Das zugehörige Steuerwort lautet: 10010010 bin.

Mit »OUT &F7FF,&X10010010« ist diese Aufgabe erledigt. Ihnen wird dabei vielleicht aufgefallen sein, daß für die Angabe des Registers C zwei Definitionen (Bits) nötig sind. Dies hat mit den bereits erwähnten anderen Betriebsarten des 8255 zu tun. Wenn Sie die Datenflußrichtung des 8255 definieren, setzen Sie immer beide Halbports dieses Registers gleich. Wie gehen wir nun konkret vor, wenn wir Daten aus einem Port einlesen oder an diesen ausgeben wollen? »OUT &F6FF, &x11001100« gibt beispielsweise den Wert 11001100 bin auf Register C aus. »A=INP(&F5FF)« liest die aktuell am Port B anliegenden acht Datenbit ein. Damit sind wir nun endlich in der Lage eine direkte Abfrage des Busy-Eingangs auch in Basic zu realisieren. Das Busy-Signal, das wir am Druckerport auf Leitung 11 eingeben, wird nämlich



Im CPC 6128 sind die vier Draht-Brücken (LK 1 bis LK4) etwas versteckt über dem Schnittstellen-Baustein erkennbar



Deutlich erkennbar liegen die Draht-Brücken für die Einschaltmeldung des CPC 664 rechts oberhalb des »8255«

zum 8255 weitergeleitet und zwar ist es mit Port B, Datenbit D6 verbunden. Mit folgendem Experiment können Sie sich die Wirkung anschauen:

```
10 A=INP(&F5FF):PRINT BIN$(A,8):
GOTO 10
```

Wenn Sie diese Schleife laufen lassen, erhalten Sie zunächst die Ausgabe »01011010«. Das zweite Bit von links ist gesetzt, der Drucker damit empfangsbereit. Wir simulieren nun einmal einen vollen Druckspeicher. Dazu verbinden wir Leitung 11 auf der Platinenoberseite mit der Unterseite der Platine, einem Massepol. Jetzt werden Sie die folgende Ausgabe erhalten: »00011010«. Das zweite Bit von links, also D6, ist nun auf Null gesetzt. Mit

»IF (INP(&F5FF) AND &x01000000) =&x010000000 THEN«

läßt sich auf sehr einfache Art und Weise überprüfen, ob der Drucker derzeit empfangsbereit ist oder nicht.

Dieselbe Abfrage können Sie aber auch für andere 1-Bit-Eingaben verwenden. Die Eingabeeinheit muß einfach nur den Busy-Eingang auf Masse legen. Damit läßt sich beispielsweise eine Alarmanlage konstruieren. Man legt einen sehr dünnen Draht vom Busy-Eingang auf Masse, reißt dieser, verändert sich das Signal und das Programm löst einen Alarm aus.

Der Busy-Eingang ist jedoch nur eines von den insgesamt 24 Bits, die uns am 8255 für I/O-Operationen zur Verfügung stehen. Was machen nun die anderen?

Das Blockschaltbild zeigt uns, daß das IC mit einer Vielzahl von Baugruppen in Verbindung steht. Port B ist fest als Eingang programmiert. Über ihn laufen alle Abfragen, außer der Tastaturabfrage. Die einzelnen Bits haben dabei folgende Belegung: Bit 0 ist für den Benutzer von geringerem Interesse. Hier ist der Zustand des SYNC-Impulses des CRTC abfragbar. Etwas für Ästheten sind die nächsthöheren vier Bit. Durch Drahtbrücken wird hier der auf dem Bildschirm aufgezeigte Firmenname fixiert.

Wollen Sie einen Freund oder eine Freundin durch einen neuen Computer beeindrucken? Oder klingt Ihnen der Titel »Schneider« gar zu deutsch und wäre Ihnen das originale »AMSTRAD« oder vielleicht »ORION« oder »SOLAVOX« lieber? Der folgende Absatz hilft Ihnen bei dieser kleine Trickserei.

Vier Computer in einem

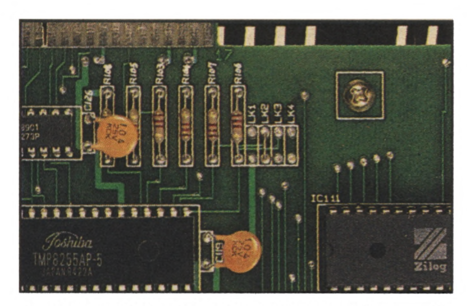
Auf der Platine befinden sich neben dem 8255 vier Plätze für Drahtbrücken. Diese sind mit LK 1 bis LK 4 bezeichnet. Jede dieser Brücken legt genau einen Eingang, eine Datenleitung, von Port B auf Masse. Der Zustand der Datenleitungen D1 bis D4 wird beim Einschalten vom Betriebssystem abgefragt und erzeugt den entsprechenden Firmennamen auf den Bildschirm. Brücke 2 veranlaßt, daß der CPC als »Schneider« erscheint. Wenn Sie nun Brücke 2 auslöten und durch eine andere Verbindung ersetzen, verändert der Computer seine Einschaltmeldung. Bei diesem Experiment sollten Sie allerdings beachten, daß mit der Änderung der Brücke, wegen des damit verbundenen Eingriffs in die Maschine, natürlich die Garantie erlischt.

Bit 5 ist mit der Exp-Leitung des »Expansion-Connectors« verbunden. Setzt man Pin 48 dieses Erweiterungsanschlusses (siehe Anhang des Bedienerhandbuches) auf Low-Pegel, so kann diese Veränderung durch Testen des Bit 5 in Kanal B festgestellt werden.

Bitweise durch den Input-/Output-Bereich

Die Abfrage geschieht dabei analog zu »Busy«. Über diesen Eingang kann ein externes Gerät dem CPC seine Existenz mitteilen. Daneben dient dieser Eingang natürlich auch zur Realisierung von Steueranwendungen am Expansion-Port oder einer seriellen Schnittstelle, wie beim Druckausgang beschrieben.





Auch im CPC 464 sind die Brücken, wie beim 664, frei zugänglich in der Nähe des I/O-Controllers zu finden

Bit 6 kennen wir bereits. Hier liegt das Busy-Signal an. Bit 7 und damit die letzte noch verbliebene Datenleitung auf diesem Port, ist mit dem Kassettenrecorder verbunden (beziehungsweise mit dem entsprechenden Pin des Steckverbinders). Über diesen Kanal liest der Computer Daten vom Recorder.

Kommen wir nun zum Port C, der im CPC fest als Ausgang definiert ist. Die unteren drei Bit steuern die Tastaturmatrix. Die Tastatur ist in zehn Reihen zu je acht Spalten angeordnet. Durch Decodierung der unteren vier Bit mit Hilfe eines BCD-Dezimaldecoders wird eine Tastaturreihe auf Low-Pegel gelegt.

Drückt man nun irgendeine Taste, so wird das Low-Signal von einer Reihe auf eine Spalte weitergeschaltet; die restlichen Spalten bleiben High. Der CPC tastet jede fünfzigstel Sekunde alle zehn Reihen durch Anlegen verschiedener Werte an diese vier Bits ab und überprüft dann, ob und welche Informationen er in den Spalten wiederfindet. So analysiert er die gedrückten Tasten.

Mit Bit 4 wird der Motor des Kassettenrecorders bedient. Bit 5 stellt die Ausgabeleitung für die Datenübergabe an den Kassettenrecorder dar. Hier wird das Tonfrequenz-Signal übertragen. Die Bits 6 und 7 dienen als Steuersignale für den Soundchip.

Verbindung für den guten Ton

Port A wird beim CPC sowohl für die Dateneingabe wie auch für die Ausgabe benutzt. Sollen Daten in den Soundprozessor geschrieben werden, so ist Kanal A auf Ausgabe programmiert. Die zu übermittelnden Daten werden in Port A abgelegt, danach wird mittels Bit 6 und 7 von Kanal C ein Steuercode ausgegeben, der den Soundprozessor zur Datenübernahme veranlaßt.

Nun zur Eingabeseite: Der Soundgenerator verfügt über einen 8-Bit-Port, der wahlweise auf Ein- oder Ausgabe programmiert werden kann. Im CPC ist er auf Eingabe programmiert. Auf diesem Kanal liegen die acht Spalten unserer Tastaturmatrix. Die mit dem Register C angewählte Reihe (vergleiche Kanal C) führt bei Tastendruck zu einem Setzen irgendeines Bits im 8-Bit-Port des programmierbaren Soundgenerators (PSG). Indem der Computer nun nacheinander alle acht Spalten durchtastet und die entsprechenden Reihen abfragt, kann er feststellen, welche Taste gedrückt wurde.

Sie sehen also, daß auch die Schneider-Computer verborgene Reize besitzen, die sich erst bei genauerem Hinsehen offenbaren. Aber die richtige Faszination der Computer entsteht nur durch diesen tiefen Einblick. Und viele Dinge sind im Basic – sei es auch noch so gut und umfangreich – einfach nicht durchführbar. Es lohnt wirklich, sich ein wenig Zeit zu nehmen und dem Schneider aufs Bit zu schauen, um bald mit ihm auf du und du zu stehen und so seine Fähigkeiten bis ins letzte auszuloten.

(Carsten Straush/ja)

Grafik maßgeschneidert



Im ersten Teil dieses Artikels (»Aus dem Schneider«) ging es um die Kommunikation zwischen CPU und Außenwelt. Jetzt geht es um die Arbeit mit dem wichtigsten Ausgabegerät, dem Bildschirm, und um die Ansteuerung der Speicherbausteine.

Werfen wir zunächst zur groben Orientierung einen Blick auf den Blockschaltplan im ersten Teil dieses Artikels, diesmal allerdings auf das obere Drittel.

Der Videoteil des CPC besteht im wesentlichen aus dem ULA und dem Videocontroller. Sie erkennen diese beiden Bausteine im oberen Teil unserer Abbildung. Quer über den Speicherbausteinen thront das ULA. Links daneben sehen Sie den Videoprozessor. Das ULA erfüllt so vielfältige Aufgaben, daß ihm im Rahmen des Systems neben der eigentlichen CPU fast die Rolle eines Hilfsprozessors zukommt. Technisch gesehen handelt es sich bei einem ULA um eine Matrix aus Widerständen, Kondensatoren und Transistoren (insgesamt mehr als 2200 Einzelkomponenten), die mit Hilfe des technischen Verfahrens der Maskenprogrammierung miteinander verbunden werden können. Das Kürzel ULA steht dabei für »Uncomitted-Logic-Array«, also unverbundene logische Gatter. Durch die Herstellung von Verbindungen ist es nun für einen Computerhersteller ohne größere Schwierigkeiten möglich, auf einem solchen Baustein zum Teil sehr verschiedenartige Funktionen zu integrieren.

Dies ist der Grund dafür, daß man beim CPC nur sehr wenige hochentwickelte Standardbausteine neben einigen Hilfschips findet, jedoch kaum nichtintegrierte Bauteile. Fast alles ist im ULA zusammengefaßt.

Das Array nimmt beim CPC im wesentlichen drei verschiedene Aufgaben wahr. In Zusammenarbeit mit dem CRTC-Controller gibt es das Bildsignal aus. Der Gatterbaustein ist dabei für die Auswahl der abzubildenden Farbe eines Punktes zuständig und stellt überdies die unterste Adreßleitung zur Auswahl des Bildschirmspeichers zur Verfügung. Der eigentliche Videochip, der CRTC (Cathod-Ray-Tube-Controller) adressiert immer zwei Byte gleichzeitig und fragt damit die in ihnen enthaltene Farbinformation ab. Welches Byte dann am Schluß bearbeitet wird, legt das ULA fest. Daneben enthält das, auch Gate-Array genannte, ULA eine Reihe von Registern, die für die Zwischenspeicherung der aktuellen Kombination von INKs, also der verwendeten Farben, benutzt werden, mit deren Hilfe der Baustein die jeweils auszugebende Farbkombination für einen Bildpunkt bestimmt.

Die zweite Aufgabe des ULA besteht in der Auswahl von Bildschirm-Modus und Speicherbereichsverteilung. Dazu verfügt es über ein weiteres Register (das MODE-Register), in dem einzelne Bits als Kennzeichen für die Speicherung des Bildschirmzustands benutzt

werden. Der Bildschirm-Modus ist für das ULA eine Kenngröße, anhand der es entscheidet, ob ein anstehendes Datenbyte als komplexe Farbinformation oder als mehrere Bildschirmpunkte zu interpretieren ist.

Der dritte wichtige Aufgabenbereich des ULA ist die zeitliche Koordination der Arbeit der anderen Bausteine. Das ULA erzeugt mit Hilfe einiger Gatter und eines Quarzes den Hauptsystemtakt von 16 Megahertz und liefert, indem es den Takt aufteilt, fast alle anderen vom System benötigten Intervalle. So zum Beispiel den Prozessor-Takt mit 4 Megahertz, die Takte für den Videocontroller und den Soundgenerator mit je 1 Megahertz. Daneben ist das ULA auch für den Hauptinterrupt-Takt zuständig, der auch als FAST TICKER bezeichnet wird. Das ist eine Unterbrechung, die 300mal pro Sekunde auftritt und im Betriebssystem für den Aufruf ständig wiederkehrender, schneller Ereignisse, benutzt wird. Eine nochmalige Teilung durch 6 liefert einen weiteren Unterbrechungstakt, den sogenannten TICKER, der für den Aufruf von zeitlich unkritischen Routinen, wie zum Beispiel der Tastaturabfrage, verwendet wird.

Stein für Stein

Das ULA ergänzt und steuert dabei die anderen Bausteine; im Videobereich ist dies besonders der Videocontroller.

Dieser Baustein liest aus dem frei wählbaren 16 KByte großen Bereich des Bildschirm-RAM Byte für Byte die Bildinformation aus und übersetzt diese in das Bildsignal für den Monitor. Dabei ist der Videocontroller in weiten Bereichen programmierbar. So können zum Beispiel die Anzahl der Zeichen pro Zeile oder ähnliche für den Bildaufbau notwendige Daten softwaremäßig gesteuert werden. Dies geschieht, indem in interne Register des Videocontrollers geschrieben wird. Der Benutzer braucht sich mit diesen Registern normalerweise nicht auseinanderzusetzen, weil sie vom Computer bei der Initialisierung auf die notwendigen Werte gesetzt werden und eine Änderung meistens nicht sinnvoll ist. Eine Ausnahme bilden die Register 12 und 13, die die Bildschirm-Steueradresse enthalten. Für eine sinnvolle Manipulation der Register benötigt man jedoch umfangreiches Wissen über den Aufbau des Grafikspeichers.

Betrachten wir uns also, wie die einzelnen Bausteine zusammenarbeiten, um das Bild auf dem Monitor zu erzeugen, beziehungsweise, um eine korrekte Freischaltung der einzelnen Speicher zu erreichen. Beginnen wir mit dem letzteren. Worum handelt es sich bei dem Begriff »Freischaltung«?

Wir haben schon im ersten Teil dieses Artikels angerissen, daß die im CPC verwendete CPU, der Z80-Prozessor, maximal 65536 verschiedene Adressen über den Adreßbus unterscheiden kann. Dies sind genau alle Kombinationen, die sich ergeben, wenn man die verschiedenen Leitungen des Adreßbusses wahlweise auf 0 oder 1 setzt. Nun verfügt der CPC bereits über 64 KByte RAM, so daß für die Adressierung des ROM (weitere 32 KByte) eigentlich kein Adreßbereich mehr zur Verfügung steht. Man müßte sich also bei der Speicher-Auswahl des Schneider-Computers etwas einfallen lassen. Die Lösung sieht folgendermaßen aus: Beim Schreiben wird immer das RAM adressiert, da nur Inhalte von RAM-Bausteinen wertmäßig geändert werden können. Soll dagegen ein Lesevorgang ausgeführt werden, so muß entschieden werden, ob die Information aus dem ROM oder dem RAM kommen soll. Diese Auswahl übernimmt das ULA. Dazu beinhaltet es ein sogenanntes Multifunktionsregister, wo in zwei Bits gespeichert ist, ob RAM oder ROM betroffen sind. Die Speicherbausteine verfügen jeweils über einen Freigabe-(ENABLE-)Eingang. Nur wenn dieser gesetzt ist, wird ein adressierter Speicher auch aktiv. Es findet also, je nachdem womit man arbeiten will (ROM oder RAM), eine Umschaltung zwischen den beiden Speicherbereichen, das sogenannte Bank-Switching, statt. Beim CPC 464 und 664 wird das Bank-Switching nur für die Umschaltung zwischen ROM und RAM benötigt. Der CPC 6128 dagegen geht auf diesem Weg noch ein Stückchen weiter. Er kann mit dem auf der Diskette mitgelieferten Programm »Bankman« auch noch zwischen zwei verschiedenen RAM-Blöcken umschalten.

Je nach dem Inhalt seines Multifunktionsregisters wählt das ULA über die Freigabeleitung entweder RAM oder ROM aus. Ein direkter Zugriff auf dieses Register in Basic ist nicht besonders sinnvoll, da der Computer selbst teilweise mehrmals bei der Übersetzung eines Basic-Befehls auf parallel liegende Speicherbereiche zurückgreift. Wenn er beispielsweise einen Grafikbefehl wie PLOT ausführt, muß er zunächst in den Basic-Interpreter, denn dort decodiert er ja erst einmal den Befehl. Dann muß er den, parallel zum Grafikspeicher liegenden, Basic-Interpreter ausschalten, um die Änderung im Grafikspeicher auch durchführen zu können. Gegebenenfalls schaltet er danach wieder auf das obere ROM zurück. Eine Änderung unsererseits in Basic wäre daher nur von kurzer Dauer und damit absolut nutzlos.

Für den Maschinensprache-Programmierer hält die Firmware des CPC jedoch fünf Routinen bereit, die zur Umschaltung zwischen ROM und RAM dienen. Der ROM-Bereich ist in zwei Blöcke mit je 16 KByte aufgeteilt, die an der Unter- beziehungsweise Obergrenze des Adreßraums liegen. Der obere Teil des ROM enthält dabei im wesentlichen den Basic-Interpreter, während der untere Bereich die Firmware-Routinen bereitstellt. Wenn Sie ein Maschinen-Programm mit »CALL« aufrufen, ist normalerweise der untere ROM-Bereich eingeschaltet, der obere dagegen nicht. Zum Umschalten dienen folgende Routinen:

KL-U-ROM-ENABLE (schaltet oberes ROM ein und den darunter liegenden Grafik-Speicher aus) Adresse: B900 hex

KL-U-ROM-DISABLE (schaltet das obere ROM aus und gibt den darunter liegenden Grafik-Speicher zum Lesen frei) Adresse: B903 hex

KL-L-ROM-ENABLE (schaltet unteres ROM ein und den darunter liegenden Basic-Speicher aus) Adresse: B906 hex

KL-L-ROM-DISABLE (schaltet das untere ROM aus und den darunter liegenden Basic-Speicher wieder ein) Adresse: B909 hex

KL-L-ROM-RESTORE (stellt den ROM/RAM-Zustand, der vor dem letzten Wechsel bestand, wieder her. Man kann natürlich auch mit »KL-U-ROM-DISABLE« ein »KL-U-ROM-ENABLE« wieder aufheben. War das obere ROM jedoch vor dem »ENABLE« bereits eingeschaltet, wird durch diesen Aufruf der Originalzustand nicht wieder hergestellt. Besser ist daher diese Routine. Adresse: B90C hex

Multi-Talent

Mit der Freischaltung kennen Sie nun den Weg, der dem CPC überhaupt den Zugriff auf den Grafikspeicher ermöglicht und diesen vom parallel liegenden ROM trennt. Damit können wir uns den einzelnen Bereichen der Bilddarstellung zuwenden. Wir beginnen mit der Farbdecodierung.

Obwohl das ULA mit der Speicherumschaltung schon eine Menge zu tun hat, ist das Bank-Switching nur ein kleiner Teil seiner Aktivitäten. Sein Hauptaufgabengebiet ist die Bildschirmdarstellung und hier speziell die Auswahl der Farben. Wenn Sie vor Ihrem Schneider schon auf anderen Heim- oder Personal Computern gearbeitet haben, wird Ihnen beim Schneider sicher zuerst die



etwas seltsame Art der Farbwahl auffallen. Der Computer gibt die Farbwerte der einzelnen Bildpunkte nicht direkt an, sondern definiert sie indirekt mit Hilfe der sogenannten INK-Register. Insgesamt existieren 16 solcher Farbregister. Alle haben ihren Sitz im Inneren des ULA. Es liegt jedoch auch noch eine Kopie von ihnen im Speicher des CPC. Dies hat folgenden Grund: Sie wissen sicher, daß der CPC Farben blinkend darstellen kann. Wenn Sie etwa definieren »INK 1,11,6«, wechselt der Cursor ständig zwischen rot und blau. Nun ist es aber nicht so, daß der CPC Farben manchmal blinkend und dann wieder stehend definiert. Im Gegenteil. Farben werden im Schneider-Computer prinzipiell blinkend definiert. Wenn Sie also einen gelben Buchstaben auf dem Bildschirm sehen, so ist dies in Wirklichkeit ein Buchstabe, dessen Farbe ständig von »Gelb auf Gelb« wechselt. Allerdings wirkt sich dies optisch nicht aus, da statt eines wirklichen Wechsels ständig derselbe Farbwert gesetzt wird. Technisch sieht die Lösung folgendermaßen aus: Die CPU schreibt zunächst den ersten Farbwert für jedes durch »INK« definierte Register in die Farbregister des ULA. Nach einer vorgegebenen Zeitspanne werden diese Farben gegen die zweite Farbkombination ausgetauscht und die Farbregister im ULA mit den neuen Farbwerten geladen. Normalerweise geschieht dies in Intervallen von 0,2 Sekunden, das heißt jede fünftel Sekunde wird die Farbe gewechselt. Wenn Sie zwei Farbwerte für ein INK-Register eingeben, wird auf die andere Farbe umgeschaltet, ansonsten bleibt die alte Farbe erhalten. Deshalb gibt es auch keinen Unterschied zwischen INK 1,24 und INK 1,24,24. Intern verarbeitet der Computer den ersten Fall grundsätzlich so wie den zweiten. Ein Direktzugriff auf die Farbregister des ULA ist deswegen auch relativ schwierig, denn jede fünftel Sekunde erfolgt ja ein automatisches Laden der Farbregister des ULA. Um sich hier dennoch einen kurzen Einblick zu verschaffen, führen Sie folgendes Experiment durch:

Das ULA wird über die Adresse 7FFF hex als I/O-Baustein angesprochen. Um ein Farbregister zu setzen, sind zwei Schritte nötig. Zuerst muß dem ULA die Nummer des zu ändernden Registers mitgeteilt werden: »OUT&7FFF,&x0000nnnn«.

Die Unterscheidung zwischen den einzelnen Registern des ULA erfolgt bitorientiert, das heißt die obersten beiden Bits geben an, ob die Farbnummer (00) oder der Farbwert (01) übergeben, oder das Multifunktionsregister (10) angesprochen werden soll. Die durch »nnnn« bezeichneten Stellen nehmen dabei die Nummer des Farbregisters als Binärzahl auf. Sie können diesen Wert errechnen, indem Sie »PRINT BIN$(<Nummer>,4)« eingeben. Bei Farbregister (INK) 1 wäre das beispielsweise 0001, bei 2 0010 und so weiter. Nun müssen wir die auszugebende Farbe übermitteln:
»OUT&7FFF,&x010fffff«

Die fünf »f« enthalten als Binärzahl den Farbcode. Wollen Sie also zum Beispiel »INK 1,11« simulieren, muß die Befehlsfolge so aussehen:
»OUT&7FFF,&x00000001:OUT&7FFF,&x01001011«

Es geht auch ohne Basic

Dabei ist 01011 bin, der hintere Teil des zweiten Wertes, die Dezimalzahl 11. Wenn Sie diese Zeile nun eingeben, wird sich möglicherweise nichts ändern; beziehungsweise werden Sie wegen des sofortigen Wechsels nach einer fünftel Sekunde nur ein kurzes Aufblitzen der neuen Farbe wahrnehmen. Zwei Änderungen verbessern unser Experiment: Zum ersten können Sie mit »SPEED INK 255,255« die Umschaltung verzögern. Die Rückschaltung erfolgt dann erst nach 5 Sekunden. Oder aber, Sie schlagen den CPC mit seinen eigenen Waffen. Wenn Sie nämlich permanent die OUT-Kommandos in einer Schleife wiederholen, überschreiben Sie damit jeweils die Änderung des Computers und Ihre Farbwahl setzt sich durch. Die erforderliche Schleife dazu sieht so aus:
»10 OUT&7FFF,&x00000001:OUT&7FFF,&x01001011:GOTO 10«

Noch ein Tip am Schluß für die Besitzer eines Grün-Monitors. Der Effekt wird besser sichtbar, wenn Sie vorher die Schriftfarbe mit »INK 1,4« abdunkeln.

Sie wissen nun, wie die Farbdefinition beim Schneider vor sich geht. Diese Art der Farbfestlegung hat Konsequenzen für den Aufbau des Bildschirmspeichers. Während dort bei vielen anderen Computern die einzelnen Farbpunkte durch Farbcodes repräsentiert werden, stehen im Schneider-Computer hier die zu verwendenden INKs. Bis zum Grafikspeicher operiert der CPC also immer noch mit PEN und PAPER. Erst das ULA übersetzt dann die Registercodes in die eigentlichen Farben. Dies ist auch der Grund dafür, daß sich bei einer INK-Definition der gesamte Bildschirminhalt, der mit dieser Farbe beschrieben wurde, ändert. Im Bildschirmspeicher sind nur die Register abgelegt, aus denen die einzelnen Punkte dann ihren Farbwert erhalten sollen.

Der Bildschirmspeicher wird durch zwei Zeiger grob untergliedert, die Bildschirmbasis und den Bildschirmoffset (Bild 1).

Die Bildschirmbasis gibt an, in welchem Bereich des RAM sich der Bildschirmspeicher befindet. Das RAM des CPC kann, wenn wir einmal die Grundvariante betrachten (also ohne die zweite 64-KByte-Bank des CPC 6128), in vier Blöcke zu je 16 KByte aufgeteilt werden. Im untersten RAM-Block liegen die für das Starten des Betriebssystems benötigten RESTART-Anweisungen. Ein weiterer Teil des Speichers wird für die Zwischenspeicherung der aktuellen Basic-Zeile bei der Eingabe benutzt. Ebenfalls hier beginnt der Basic-Quelltext, also das noch nicht in Maschinenbefehle übersetzte Basic-Programm. Es folgt ein weiterer 16-KByte-Bereich, der, je

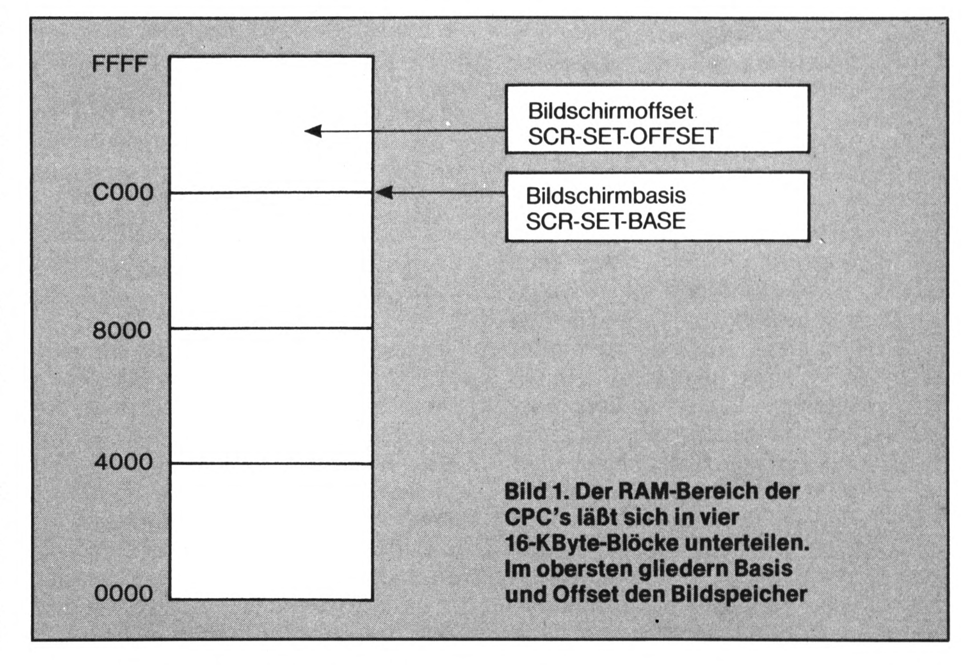


Bild 1. Der RAM-Bereich der CPC's läßt sich in vier 16-KByte-Blöcke unterteilen. Im obersten gliedern Basis und Offset den Bildspeicher

nach Ausdehnung des Basicprogramms, auch noch von diesem belegt wird, ansonsten aber frei zur Verfügung steht. Er reicht von Adresse 4000 hex bis 7FFF hex. Block 3 wird von Systemvektoren und -variaben genutzt und ist somit nicht als Bildschirmspeicher geeignet. Bei der Initialisierung definiert der CPC den Bereich zwischen C000 hex und FFFF hex, also den vierten und letzten Block, für die Speicherung der Bildinformationen.

Wie wär's mit zwei Bildschirmen?

Die Wahl des Bildschirmspeichers verläuft über den Video-Chip. Dafür stellt das Betriebssystem auf Adresse BC08 hex die Routine SCR-SET-BASE bereit. Beim Aufruf dieser Routine muß der Akkumulator, Register A der CPU, das signifikante Byte der Speicheradresse enthalten.

Signifikantes Byte bedeutet, daß für die Auswahl nur die obersten beiden Bit der übergebenen Adresse benutzt werden. Wir müssen also lediglich das High-Byte der Anfangsadresse des ausgewählten Speicherbereiches an das Betriebssystem übergeben. Für Block 2 ist A mit 40 hex zu laden, denn dieser Bereich beginnt ja ab Adresse 4000 hex. Wir wollen unser Wissen gleich einmal anhand eines einfachen Beispiels testen. Zuvor beschäftigt uns jedoch noch eine weitere Routine, die für die Organisation innerhalb des Grafikspeichers zuständig ist. Mit SCR-SET-OFFSET (Adresse BC05 hex) wird der obere linke Eckpunkt des Bildschirms festgelegt. Wir haben es also beim CPC nicht mit einem festen Speicher zu tun, bei dem Bildpunkte genau definierten Speicherstellen zugeordnet sind. Um die Wirkungsweise dieser beiden Routinen klarzumachen, gibt es nur eines: Ausprobieren.

Eine sinnvolle Anwendung für »SCR-SET-BASE« ist die Definition eines zweiten Grafikspeichers. Wir haben aber bereits einen Grafikspeicher. Wozu also einen zweiten? Nehmen Sie einmal an, Sie wollten ein Zeichenprogramm schreiben, das über den ganzen Bildschirm malen kann. Wo wollen Sie nun Erklärungen und Anweisungen oder auch nur ein einfaches Menü, das die Strichstärke ihres Pinsels festlegt, ausgeben. Schon wenn Sie nur ein einziges Zeichen auf dem Bildschirm ausgeben, zerstört es die gerade so mühsam erstellte Grafik. Oder stellen Sie sich vor, daß Sie in einem Programm, das verschiedene Bilder in hochauflösender Grafik benutzt (beispielsweise einem Spiel), durch eine einzige kurze Anweisung auf den anderen Bildschirm umschalten und damit fast ohne zeitliche Verzögerung ein neues Bild zur Verfügung haben. Nicht zuletzt wäre es auch denkbar, im Grafikspeicher Nummer 1 ein fehlerhaftes Listing parat zu halten, während man parallel dazu in Grafikspeicher 2 das Ergebnis des Programmlaufs betrachtet. Genügend Anwendungen für einen zweiten Grafikspeicher gibt es also. Überlegen wir also, wie wir ihn mit Hilfe der Speicherumschaltung realisieren können.

Erster Grafikspeicher soll der normale Bereich von C000 hex bis FFFF hex bleiben. Als zweiten Block sehen wir den Bereich von 4000 bis 7FFF hex vor.

Geben Sie »POKE &D700,&FF« ein, und Sie erhalten, etwa in der Mitte des Bildschirms, einen gelben Strich. Als nächstes tippen Sie den folgenden Vierzeiler ein:

```
10 MEMORY &3FFF
20 DATA 3e,40,cd,08,bc,c9
30 FOR i=42000 TO 42005:READ
   a$:POKE i,VAL("&"+a$):NEXT
40 CALL 42000
```

Zunächst setzt dieses Programm die Speicherobergrenze auf 3FFF hex herab, um Kollisionen mit dem Basic, speziell den Stringvariablen, zu vermeiden. Danach wird ein kleines Maschinen-Programm in den Speicherstellen 42000 bis 42005 erzeugt und aufgerufen.

Dieses wiederum lädt den Akkumulator mit dem Wert 40 hex, ruft SCR-SET-BASE auf und kehrt wieder ins Basic zurück. Löschen Sie jetzt den Schirm mit »CLS« und setzen Sie dann ein paar Punkte mit dem oben benutzten »POKE«. Nichts passiert. Dies ist auf den ersten Blick natürlich verwunderlich. Wenn wir das Ganze durchdenken, so wird der Sachverhalt aber relativ schnell klar. Durch die Verschiebung der Bildschirmbasis liegt der Punkt in der Mitte des Bildschirms nicht mehr bei D700 hex, sondern 32 KByte tiefer, auf der Adresse 5700 hex, so daß ein »POKE« an diese Speicherposition wieder den gewünschten Strich erzeugt.

Hin und her

Ändern Sie nun in Zeile 20 die »40« auf »F0« ab und lassen das Programm laufen, so erhalten wir nochmals die ursprüngliche Bildschirmanzeige. Auf diese Weise läßt sich auf zwei völlig voneinander unabhängigen Bildschirmen arbeiten. Veränderungen im »Neben«-Bildspeicher bei eingeschaltetem »Haupt«-Bildspeicher sind aber nur durch direkten Speicherzugriff möglich.

Wir müssen uns daher mit der Speicherstruktur etwas näher befassen. Wichtig dabei ist erst einmal die genaue Kenntnis der Routine SCR-SET-OFFSET und der damit zusammenhängenden Hardwaregrundlagen. Wir hatten ihre grundsätzliche Bedeutung bereits angerissen. Sie legt den Anfangspunkt des Bildschirms (also die linke obere Ecke) im Speicher fest. Dies geschieht über die Änderung von zwei Registern im CRTC (den Registern 12 und 13). Vor dem Aufruf der Routine müssen wir in das Registerpaar HL den Wert des neuen Offsets schreiben. Die obersten beiden Bit werden dabei vernachlässigt.

Bevor wir uns mit einer Änderung des Offset-Zeigers beschäftigen, sollten wir uns erst einmal klarmachen, wozu er genau dient. Eigentlich wäre es viel einfacher, wenn dieser Punkt als erste Stelle im Bildschirmspeicher, beispielsweise als Byte C000 hex oder 4000 hex festgelegt wäre. Um der Sache auf den Grund zu gehen, machen wir am besten einige Experimente.

Video-Genie

Dafür greifen wir direkt auf den Videoprozessor beziehungsweise seine beiden Register zu. Der CRTC läßt sich über vier Adressen ansprechen, wovon allerdings nur zwei sinnvoll nutzbar sind. Zunächst muß über das Adreßregister des CRTC die Nummer des internen Datenregisters gewählt werden, bevor man dieses dann mit Daten füttern kann.

Das Adreßregister erreichen wir unter der Adresse BCXX hex, wobei die »XX« bedeuten, daß diese beiden Stellen nicht benötigt werden.

»OUT&BCFF,13« teilt dem Videogenerator mit, daß wir auf sein internes Register 13 zurückgreifen wollen; die unteren, niederwertigen 8 Bit des Offset-Zeigers. Über die Adresse BDFF hex können wir nun Daten an das ausgewählte Datenregister senden.

Normalerweise sollten Sie diesen Direktzugriff mit OUT nur in absoluten Ausnahmefällen benutzen, weil keine Rückmeldung an das Betriebssystem erfolgt. Da der CPC aber für die Organisation des Bildschirms auf die genauen Werte für Basis und Offset angewiesen ist, ist höchste Vorsicht geboten. Bei den nachfolgenden Experimenten sollten Sie daher auch kein wichtiges Programm mehr im Speicher haben; es könnte sonst ein Opfer Ihrer Experimente werden.

Der CPC verwendet den Offset-Zeiger für den sogenannten Hardware-Scroll. Tippen Sie einmal das folgende kleine Programm ein:

Fortsetzung auf Seite 26

Massenspeicher der Zukunft

CD-ROM

Ein Wunder an Speicherkapazität

auf kleinstem Raum!

Für wenig Geld ein halbes Gigabyte (das wären z.B.: sämtliche Daten einer 26-bändigen Enzyklopädie) bieten die ersten CD-ROMs. Auch Festplattenlaufwerke werden immer preiswerter. Die Happy-Computer-Mai-Ausgabe vermittelt Ihnen das technische Know-how und sagt im Detail was der Markt bietet.

2. großes Thema — Programmiersprachen: Mit welcher Sprache beginnen? Dazu eine umfassende Marktübersicht und hilfreiche Tips.

3. großes Thema — 68000er-Prozessor: Eine Übersicht, Vergleiche und Tips helfen Ihnen bei der Wahl Ihres Computers mit diesem Prozessortyp.

Außerdem finden Sie Tips, Tricks und Hilfen im Umgang mit CP/M-Programmen — Den neuen Spectrum 128 im Test — Einen großen excellenten Spieleteil mit Tips, Tricks und Trends — Eine Hardware-Bastelei, mit der Sie Ihren User-Port »verdoppeln« können und den Wetterbericht mal ganz privat: Meteosat mit Atari ST.

Das Listing des Monats heißt »Toolbasic 1.1«: Sehr hohe Bildschirmauflösung (640 x 400 Bildpunkte) für alle Schneider-Computer.





erhalten Sie Mitte jedes Monats bei Ihrem Zeitschriftenhändler. Die Mai-Ausgabe erscheint am 7. April 1986.



Gutschein

FÜR EIN KOSTENLOSES PROBEEXEMPLAR VON HAPPY COMPUTER

JA, ich möchte »Happy-Computer« kennenlernen.
Senden Sie mir bitte die aktuellste Ausgabe kostenlos als Probeexemplar. Wenn mir »Happy-Computer« gefällt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte »Happy-Computer« dann regelmäßig frei Haus per Post und bezahle pro Jahr nur DM 66,— statt DM 72,— Einzelverkaufspreis (Ausland auf Anfrage).

Vorname, Name

Straße — PLZ, Ort

Datum — 1. Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse widerrufen kann und bestätige dies durch meine zweite Unterschrift. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs.

Datum — 2. Unterschrift

Gutschein ausfüllen, ausschneiden, in ein Kuvert stecken und absenden an: Markt&Technik Verlag Aktiengesellschaft, Vertrieb, Postfach 1304, 8013 Haar

HCS486

```
10 CLS:LOCATE 10,10:PRINT"PRO-
   BETEXT"
20 FOR i=1 TO 3
30 OUT&BCFF,13:OUT&BDFF,i*80
40 FOR t=1 TO 500:NEXT t
50 NEXT i
```

Nach »RUN« wird zunächst der Bildschirm gelöscht und »Probetext« in der Mitte des Schirms ausgegeben. In der Schleife wird der untere Teil dieses Zeigers in 80er-Schritten erhöht. Was Sie jetzt sehen, ist der Hardware-Scroll, wie ihn der CPC immer dann anwendet, wenn der Editor auf der letzten Textzeile angelangt ist und neuen Platz benötigt. Eine Verschiebung des Bildschirmspeicheranfangs um 80 Byte erzeugt also Scrolling um eine Zeile. Wie sieht es aber aus, wenn wir den Offset-Pointer nur um einen geringeren Betrag verändern, beispielsweise in 2-Byte-Schrit-

Schiebung

ten? Ändern Sie die Gesamtzahl der Schritte in Zeile 20 auf beispielsweise 40 und ersetzen den Multiplikator »80« in Zeile 30 durch eine 2. Der Text wandert nun langsam nach links. Wenn Sie dieses Spiel nun mit mehreren versetzten Texten wiederholen, wird irgendwann der Text mitten im Bildschirm abgehackt und an einer ganz anderen Stelle fortgesetzt.

Zur näheren Erklärung dieses Phänomens macht man sich am besten den Aufbau des Bildschirm-RAM mit Hilfe einer kleinen Grafikroutine klar. Versetzen Sie den Computer mit <CTRL> <SHIFT> <ESC> in den Ausgangszustand und tippen Sie nach »CLS« die folgende Zeile ein:

»FOR i=&C000 TO &FFFF:POKE I,&FF :LOCATE 1,1:PRINT HEX$(i):NEXT«

Sie lädt den Bildschirmspeicher ab Speicherstelle C000 hex mit dem Wert FF hex. Wenn Sie sich den Ablauf anschauen, werden Sie schnell die Speicherstruktur erkennen. Das Schreiben beginnt in der linken oberen Ecke, wie nicht anders zu erwarten war, da der Offset nach dem Einschalten immer 0 ist. Es werden jetzt rote Linien untereinander auf dem Bildschirm gezeichnet, und zwar für jede Textzeile zuerst die oberste von acht Bildschirmzeilen.

Der Wert in der linken oberen Ecke gibt dabei die gerade beschriebene Speicherstelle an. Bei dem Wert C800 hex, beginnt der CPC mit der Ausgabe der zweiten Bildschirmzeile. Bei D000 hex folgt die nächste und so weiter. Der Abstand entspricht also immer 2048 (800 hex).

Jede Bildschirmzeile besteht aus 640 Bildpunkten, so daß wir für ihre Speicherung jeweils 80 Byte belegen.

Pixel	MODE 0	MODE 1	MODE 2
Linkes Pixel	Bits 1,5,3,7	Bits 3,7	Bit 7
.			Bit 6
.		Bits 2,6	Bit 5
.			Bit 4
.	Bits 0,4,2,6	Bits 1,5	Bit 3
.			Bit 2
.		Bits 0,4	Bit 1
Rechtes Pixel			Bit 0

Tabelle. Organisation der drei Bildschirm-Modi

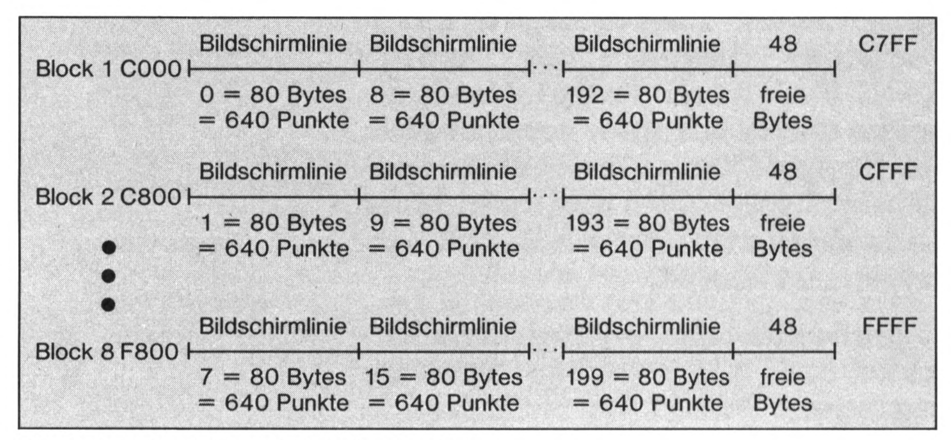


Bild 2. Adreßverteilung im Grafikspeicher

25 Textzeilen zu je 80 Byte für die Darstellung einer Bildschirmzeile ergeben einen Block von 2000 Byte, der alle obersten Bildschirmzeilen beziehungsweise alle zweiten Bildschirmzeilen und so weiter enthält (vergleiche Bild 2). Nun wissen Sie auch, warum die Verschiebung des Bildschirm-Offsets um 80 Byte ein Scrollen um genau eine Zeile bewirkt, beziehungsweise, warum es bei einem Scrollen um zwei oder zehn Byte zu den beschriebenen seltsamen Zerstückelungseffekten kommt.

Wo sind die Bytes?

Dies sind jedoch nicht die einzigen Besonderheiten des Bildschirmaufbaus. Wir haben es ja schließlich bei unserem Bildschirmspeicher nicht mit 16000 Byte zu tun, sondern mit 16 KByte (16384 Byte). Das führt dazu, daß nach dem Ende jedes 2000-Byte-Blocks 48 Byte frei bleiben. Sie können das verfolgen, indem Sie kurz vor dem Sprung von einer Bildschirmzeile zur nächsten auf den Zähler in der linken oberen Ecke schauen. Er läuft 48 Byte weiter, ohne daß auf dem Bildschirm eine Änderung eintritt. Erst nach Überschreiten der 2048er-Grenze tut sich wieder etwas auf dem Monitor. Nun könnte man meinen, hier sei eine Menge ungenutzter Speicherplatz, der nur darauf wartet, entdeckt und (zum Beispiel zur Ablage von Maschinenprogrammen) genutzt zu werden. Leider ist dies aber nicht der Fall. Denn mit jeder Änderung des Bildschirmanfangs über den Offset ändert sich auch der Anfang der acht 2000-Byte-Blöcke und damit liegt der Überhang von 48 Byte an einer anderen Stelle im Speicher.

Nach so viel Schieberei und Überlappung sollte man eigentlich denken, daß kaum eine Steigerung mehr möglich ist. Weit gefehlt! Mit der internen Speicherung der Farben für die einzelnen Bildpunkte setzt der CPC noch einen drauf.

Im Bildschirm-Modus 2 ist alles noch relativ einfach. Hier entspricht jedes Bit einem gesetzten Bildpunkt. Wenn Sie also einen Bildpunkt in diesem Modus durch Direktzugriff ändern wollen, so müssen Sie nur seine Position im Speicher mit Hilfe des Bildschirm-Offsets berechnen. Greifen Sie aber für die praktische Arbeit auf die Firmware-Routinen zurück, da diese die notwendigen Verschiebungen bereits durchführen, Ihnen also viel Arbeit ersparen.

Etwas schwieriger gestaltet es sich dagegen, wenn wir in den beiden anderen Modi operieren möchten. Die besseren farblichen Möglichkeiten werden dabei durch gleichzeitiges Setzen mehrerer Bildpunkte erreicht. So werden zum Beispiel im Modus 1 immer zwei Bildpunkte nebeneinander auf dieselbe Farbe gesetzt. Da man nun nicht mehr für jeden Bildpunkt ein eigenes Bit zur Speicherung braucht, können jetzt in zwei Bit die möglichen vier Farben codiert werden. Steht in den beiden Bit 00, so wird Ink 0 benutzt, beim Wert 01 dagegen Ink 1.

Nun wäre es schön und eigentlich

Bild 3. Gewöhnungsbedürftig: Farbcodierung im CPC

auch logisch, anzunehmen, daß Bit 6 und 7 zusammengefaßt die Farbinformationen für die beiden linken Bildpunkte liefern, Bit 5 und Bit 4 für die nächsten beiden Bildpunkte (Pixels) und so weiter. Doch was einfach wäre, ist beim CPC noch lange nicht natürlich. Die Tabelle zeigt die wahre Zuordnung der verschiedenen Bits zu den Bildpunkten. Da die Farbcodierung auf den ersten Blick etwas konfus wirkt, wollen wir einige Beispiele durchexerzieren.

Im Modus 0 soll folgendes Farb-Byte vorliegen (in binärer Schreibweise): 01010101. Als Farben ergeben sich nun für das linke Pixel Ink 0 und für das rechte Pixel Ink 15 (vergleiche Bild 3). Vollziehen Sie es einmal in Gedanken nach!

Nun soll eine grüne Strichellinie auf dem Bildschirm dargestellt werden. Dazu werden jeweils im Modus 1 vier Bildpunkte gesetzt, und danach vier Bildpunkte ausgelassen. Zunächst wird mit »INK 3,21« als Vordergrundfarbe Grün bestimmt. Wir gehen einmal davon aus, daß die linken und die rechten zwei Bildpunkte auf Hintergrundfarbe, die mittleren dagegen auf Vordergrundfarbe gesetzt werden sollen. Somit ergibt sich der Byte-Wert 01100110 bin.

Wir nehmen eine Länge von 20 Zeichen für die Linie (beginnend mit dem zehnten Zeichen) in der dritten Zeile des vierten 2000-Byte-Blocks.

Jetzt rechnen wir ein wenig. Der vierte Block beginnt bei Adresse D800 hex (dezimal 55296). Wenn wir ein Offset von Null zugrundelegen, kommen jetzt 160 Byte für die beiden Zeilen, die wir überspringen müssen, sowie weitere 30 Byte, damit unser 20 Byte langer Strich in der Mitte steht. Als Startadresse für unsere Linie erhalten wir so 55486 und damit folgende Programmzeile:

»INK0,0:INK3,21:FOR i=55486 TO 55506:POKE i,&X01100110«

Probieren Sie nun einmal, die Linie selber auf andere Farben zu setzen oder eine senkrechte Linie oder ein Rechteck auf dem Bildschirm mit Direktzugriff zu zeichnen. Eines müssen Sie dabei aber beachten. Das Verschieben von Textzeilen beim CPC geschieht durch eine Manipulation des Offset-Zeigers, vorausgesetzt man arbeitet auf dem Gesamtbildschirm.

Hard- und Soft-Scroll

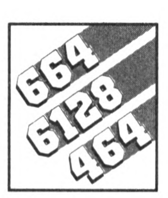
Mit jedem Scrollen verändern sich damit die Anfangskoordinaten des Bildschirms. Die einzige Möglichkeit, dies zu umgehen, ist, den Bildschirm in zwei Windows aufzuteilen. Zum Beispiel könnte man die Zeilen 1 bis 12 und 13 bis 25, als Window 0 beziehungsweise Window 1 definieren. Dadurch wird der CPC zum sogenannten Software-Scrolling gezwungen. Bei dieser Art des Scrollings tauscht der Computer den Inhalt zweier Speicherbereiche gegeneinander aus, beziehungsweise füllt er den betreffenden Bereich mit INK 0 (Hintergrundfarbe).

Das führt zwar bei Bildschirmbewegungen zu erheblichem Geschwindigkeitsverlust (vergleichen Sie einmal die Ausführungsgeschwindigkeit eines »LIST« auf dem Gesamtbildschirm mit »LIST« in einem Window), hat dafür aber den Vorteil, daß der Wert des Offset-Zeigers konstant bleibt. Definieren Sie also direkt nach dem Einschalten des Computers auf diese Weise zwei Fenster, bleibt die linke obere Bildschirm-Ecke stets an der Position C000 hex.

Sie sehen: Der CPC muß ganz schön »schuften«, um kunstvolle Grafik auf den Bildschirm zu zaubern. Wie Sie gesehen haben, ergeben sich aber auch, beispielsweise mit dem Hardware-Scroll, immense Möglichkeiten. Experimentieren Sie daher ruhig ein wenig. Es lohnt sich!

(Carsten Straush/ja)

Basic-Programm: Stück für Stück



Meist sind die Erklärungen zu Basic-Programmen so dürftig, daß man als interessierter Einsteiger nicht viel lernen kann. Wir zeigen Ihnen deshalb Befehl für Befehl, wie man auf dem Schneider programmiert.

Als Lernbeispiel für Basic dient uns ein kleines Spiel, an dem Sie sehen können, welche grundsätzlichen Probleme es beim Programmieren geben kann. Hier lernen Sie, wie man sie beheben kann und zwar Schritt für Schritt. Das vollständige Spiel finden Sie nebenstehend als Listing. Damit Sie aber etwas lernen können, sollten Sie das Programm nicht nur eintippen und erst dann weiterlesen, sondern »step by step« mit jeder besprochenen Zeile diese erst dann eintippen. Da unser Programm weitgehend linear aufgebaut ist – was man zuerst immer machen sollte – steht am Programmanfang das, was zuerst ausgeführt werden soll.

Jedes etwas umfangreiche Programm, das auch von Außenstehenden benutzt werden soll, muß über eine Erklärungsroutine (einen Programmteil, der sagt, was das Programm so alles kann) verfügen. Normalerweise benötigt ein einfaches Spiel keine großartigen Erklärungen. Die Einleitung dient hier eher als Beispiel, wie man solche Texte auf den Bildschirm bringen kann. Außerdem erspart sie dem Schreiber dieser Zeilen die Arbeit, Ihnen zu erklären, wie das Spiel funktioniert. Tippen Sie also zuerst einmal die Zeilen 10 bis 190 ein und lassen Sie dann das Rumpf-Programm laufen.

Am Anfang finden Sie einige REM-Zeilen. Neben dem Basic-Befehl REM kennt der Schneider ein Kurzzeichen für diesen Befehl, das Apostroph (»'«). Hinter diesem kann man beliebige Texte speichern. Diese werden vom Computer immer ignoriert. Für den Benutzer bilden sie aber eine Orientierungs- und Erinnerungshilfe. Damit kann man ein Programm strukturieren und erkennt an der Überschrift sofort, um welches Listing es sich handelt. Besonders bei mehreren ähnlichen Routinen, beispielsweise geänderten Programmversionen, ist das sehr nützlich. Man erkennt gleich am Anfang, was für ein Listing man eigentlich vor sich hat. Da der Computer bei REM-Markierungen nichts macht, können Sie diese natürlich auch weglassen. Das Programm wird dadurch aber unübersichtlicher.

In Zeile 110 stoßen Sie auf den Befehl »MODE 1«. Da der Schneider in verschiedenen Bildschirmmodi arbeiten kann (MODE 0=20 Zeichen pro Zeile, MODE 1=40 Zeichen pro Zeile, MODE 2=80 Zeichen pro Zeile) ist es immer sinnvoll am Programmanfang definierte Ausgangsbedingungen zu schaffen. Dies ist deshalb wichtig, da farbliche Kombinationen und auch die Zeichenpositionen (-koordinaten) von diesen Rahmenbedingungen abhängen.

Als nächstes wird der Einleitungstext mit einer Reihe von PRINT-Befehlen ausgegeben. Dazwischen eingefügte

PRINTs ohne nachfolgenden Text sorgen für den notwendigen Zeilenabstand und damit für einen interessanten Bildschirmaufbau. In Zeile 150 sehen Sie dann, wie man auch Zeichen, die nicht direkt über die Tastatur verfügbar sind, auf den Schirm bekommt. Die Symbole für die Cursortasten werden mit »+« und mit Hilfe der CHR$(x)-Funktion in die Ausgabezeichenketten eingebunden. Wir werden CHR$ in Folge noch häufiger verwenden. Dieser Befehl bringt ein Zeichen mit einer angegebenen Nummer (durch das x symbolisiert) - dem Zeichencode - auf den Monitor. Die Zuordnung von Zeichen und Codes finden Sie im Anhang des Handbuches, wo die einzelnen Zeichen (mit ihren Nummern) abgebildet sind.

Will man einen längeren Text ausgeben, zum Beispiel über mehrere Bildschirmseiten, muß man irgendeine Abfrage einbauen, damit der Betrachter zur nächsten Seite gelangt. Eine automatische Weiterschaltung nach einer festgelegten Zeit ist dabei nicht empfehlenswert, da jeder Mensch unterschiedlich lange zum Lesen braucht. Der eine müßte warten, während der andere noch nicht bis zum Ende vorgedrungen ist. Das Optimum bildet eine einfache Abfrage, bei der nur eine Taste gedrückt werden muß, um zur nächsten Seite zu gelangen. Sinnvollerweise benutzt man dazu eine möglichst große Taste, beispielsweise die SPACE-Taste. Die Anweisung dazu wird in Zeile 180 ausgegeben. Die Überprüfung steht in der nächsten Zeile. Mit INKEY$ wird dabei die gerade gedrückte Taste in die Variable z$ gelesen. Wenn keine Taste gedrückt wird oder eine andere als die Leertaste, dann steht in z$ entweder nichts ("") oder etwas anderes als ein Leerzeichen (" "). Der Schneider bleibt sozusagen ewig in dieser Zeile hängen, zumindest solange, bis die richtige Eingabe erfolgt.

Mit dem ersten »RUN« haben Sie jetzt den Einleitungstext gelesen und wissen in groben Zügen, worum es geht. Hier noch ein paar zusätzliche Informationen: Die eigentliche Spielhandlung besteht darin, durch Abschießen eines Balles, die aus dem ganzen Bildschirm verstreuten Freßwaren (durch Herzchen gekennzeichnet) »abzuschießen«, damit Ihr lieber Gegner, ein nicht allzu großes Monster, Sie nicht fressen wird. Mit einer Art Armbrust, im Spiel durch einen Strich symbolisiert, die Sie in der untersten Bildschirmzeile seitlich bewegen können (mit Cursor rechts und Cursor links) und deren Abschußrichtung (nach rechts mit SHIFT und Cursor rechts, beziehungsweise links mit Cursor links und SHIFT) Sie beinflussen können, schießen Sie Ihren Ball ab. Mit Druck auf COPY fliegt der Ball los. Seine Flugbahn ist nicht ganz einfach, da sich neben der Nahrung noch andere Gegenstände, die als Hindernisse wirken, auf dem Bildschirm befinden und den Ball unvorhersehbar reflektieren. Diese Hindernisse werden durch einfache Rechtecke symbolisiert.

Soweit zur Spielhandlung. Doch leider beginnt ein Programm nicht mit den Höhen der Programmtricks, sondern in den Niederungen von Farbdefinition und Bildschirmaufteilung. Die dazu notwendigen Zeilen finden Sie bis Zeile 280. Nicht nur der Bildschirm muß in einer definierten Ausgangsposition sein, sondern auch die Werte der verschiedenen Variablen. Dazu dient der Befehl CLEAR. Er setzt alle Variablen auf Null. Es folgt ein weiteres Initialisierungskommando: DEFINT. Da wir in unserem Spiel ausschließlich ganze Zahlen benutzen, definieren wir speicherplatzsparende Integer-Variablen (Ganzzahlen). Wozu sollen wir Speicherplatz und Schnelligkeit verschwenden? Mit DEFINT werden alle Variablen als Integer-Variablen definiert. Eine Integer kann Werte zwischen -32768 und 32767 annehmen und benötigt dabei deutlich weniger Speicherplatz als die normalen (Real-)Variablen. Sie können vom Computer bedeutend schneller verarbeitet werden.

Erst die Arbeit

Als nächstes folgen einige Bildschirmbefehle. Der Bildschirm soll in seiner Breite etwas verringert werden, um neben dem eigentlichen Spielfeld noch Erklärungen ausgeben zu können. Dazu definieren wir den Hauptbildschirm als WINDOW # 0 mit einer Breite von 35 Zeichen (Spalte 1 bis Spalte 35), und ein weiteres WINDOW # 1 mit den Spalten 36 bis 40. Die Höhe beider Windows beträgt dabei 25 Zeilen (die volle Bildschirmhöhe). Nun brauchen unsere Bildschirmfenster (die Windows) eine Farbdefinition, damit der Computer weiß, mit welcher Farbe er in den beiden Fenstern schreiben soll. Dazu müssen wir vier Farbregister definieren. Dazu dient das Kommando INK. Register 0 wird mit schwarz (Farbe 0), Register 1 mit gelb (24), Register 2 mit blau (10) und Register 3 mit rot (Farbe 6) belegt. Den Rand setzen wir ebenfalls auf blau (10). Wenn wir nun mit PAPER und PEN den beiden Windows ihre Register für die Hintergrund- beziehungsweise Vordergrundfarbe zuweisen, und dabei die Hintergrundfarbe für Window zwei aus Register 2 entnehmen (»PAPER # 1,2«), so wird beim nächsten Löschen der Bereich dieses Fensters mit der »PAPER«-Farbe beschrieben. In Window # 1 sollen später der Spielstand und die Punkte ausgegeben werden. Mit »CLS« löschen wir gleichzeitig beide Windows.

Mit der Definition der Anfangswerte für den Bildschirm sind wir fertig. Aber unser Programm braucht nun auch noch ein paar Variablen. Diese müssen zu Anfang gesetzt werden. Wir haben gesagt, daß verschiedene Gegenstände auf dem Bildschirm postiert werden sollen. Nun ist dies zwar eine für einen Menschen ausreichende Beschreibung, der Schneider benötigt aber die Informationen in anderer Form. Eine Möglichkeit dazu ist ein Datenfeld - ein sogenannter Array.

Ein solches wollen wir als nächstes definieren. Es soll zwei Koordinaten enthalten. Die X-Koordinate soll von 1 bis 40, entsprechend den 40 Bildschirmspalten, laufen. Die Y-Koordinate soll angeben, zu welcher Zeile der gespeicherte Wert gehört. In Wirklichkeit läuft das Feld über 41 Spalten (0 bis 40) und 26 Zeilen (0 bis 25). Aber unser zu groß definiertes Feld ist übersichtlicher als das kleinere.

Die Abbildung eines Zeichens, beispielsweise eines Freßsymbols auf dem Bildschirm, geschieht in zwei Stufen. Einerseits wird das Zeichen mit Hilfe von LOCATE an einer bestimmten X,Y-Position auf dem Schirm ausgegeben, zum anderen wird unter derselben Koordinate ein Eintrag im Datenfeld o(40,25) gemacht. Dieser zeigt an, ob sich an dieser Stelle ein Freßsymbol befindet oder nicht. Um zu sehen, wie dies in der Praxis erfolgt, sollten Sie nun das Programm bis einschließlich Zeile 440 eingeben und dann laufen lassen.

Jedes Array muß dimensioniert werden. Dazu dient der DIM-Befehl. In den folgenden beiden Zeilen werden weitere Variablen auf festgelegte Anfangswerte gesetzt. Dann erfolgt unser Spielbildschirmaufbau. Zuerst werden die Hindernisse definiert. Die Position ist dabei dem Zufall überlassen. Insgesamt sollen 40 Barrieren auf dem Bildschirm verstreut werden. Dies geschieht in einer Schleife. Mit RND(1) bestimmen wir eine Zufallszahl zwischen 0 und 1. Die Multiplikation mit 33 - beziehungsweise 21 - und Hinzuzählen von 2 liefert uns Werte zwischen 2 und 35 - beziehungsweise 2 und 23. An die so entstandenen Zufallskoordinaten (a,b) setzen wir mit LOCATE und CHR$ das Rechteckzeichen (Zeile 380). Gleichzeitig wird in dem Array o(a,b) die entsprechende Position mit 2 besetzt (Programmzeile 370). Anhand der 2 stellt der Computer später bei gegebenen Koordinaten fest, ob an dieser Stelle ein Hindernis steht oder nicht.

Als nächstes kommen die Futtermittel für unser Tierchen. Davon gibt es etwas mehr als von den Hindernissen (60 statt 40), so daß die FOR-TO-Schleife in Zeile 400 einen höheren Grenzwert besitzt. Die Wertzuweisung und der Ausdruck auf dem Monitor funktionieren analog, nur daß wir das Zeichen 228 (Programmzeile 430) an der mit LOCATE a,b ausgewählten Koordinate setzen. Zur Unterscheidung von den Hindernissen wird eine 1 in o(a,b) eingetragen. Die Elemente des Arrays o können also drei Zustände annehmen. Ist bei den Koordinaten x,y o(x,y)=0, so ist die betreffende Position auf dem Bildschirm leer. Bei o(x,y)=1 befindet sich an dieser Stelle ein Freßsymbol, bei o(x,y)=2 ist es ein Hindernis für den Ball.

Der Bildschirm nimmt Form an

Sie sollten nun wieder ein paar Zeilen weiter eintippen – bis einschließlich 540 – damit Sie den Rest des Bildschirmaufbaus zu Gesicht bekommen. Es handelt sich dabei um die Ausgaben im rechten Teil des Bildschirms – Punktestand und Schußzahl. Insgesamt haben Sie drei Versuche pro Spiel, in denen Sie versuchen müssen, möglichst viele »Freßherzchen« zu treffen. Die Punkte werden dabei in der Variablen »pu« gezählt. Je Freßsymbol gibt es einen Punkt. Der Punktestand wird laufend ausgegeben. Zeile 450 gibt mit einem weiteren LOCATE, diesmal auf die rechte Seite des Schirms (im Window #1), den Text »Pkte.« und drei Zeilen darunter »Schuß« aus. Da wir das »ß« nicht auf der Tastatur finden, müssen wir auf einen Trick zurückgreifen, den wir schon vom Einleitungstext her kennen – die CHR$-Funktion. Mit dieser können wir auf alle Zeichen im Zeichensatz des CPC zurückgreifen, so auch auf das griechische Beta mit der Symbolnummer 177, das dem »ß« so ähnlich sieht, daß wir es hier problemlos verwenden können. Es wird an das »Schu« angehängt und dann positionsrichtig ausgegeben. Mit Zeile 460 endet dann auch der Vorbereitungsteil. Wir gelangen zur eigentlichen Spielhandlung.

Dazu müssen wir allerdings zunächst einige Variablen erklären. Zwei Koordinaten sind für das Spiel von Bedeutung:
- die aktuelle Position des Abschußgerätes (gespeichert in xf und yf),
- die aktuelle Position des Balles, (gespeichert in x und y).

Daneben muß natürlich auch noch die Abschußrichtung vorgegeben sein. Diese wird in »ri« abgelegt. Die Anfangswerte dieser Variablen werden in den zwei Zeilen 330 und 340 im Vorbereitungsteil gesetzt.

In unserem Hauptbildschirm fehlt noch das Abschußgerät. Seine Position ist durch xf, yf bestimmt. Indem wir diese Koordinaten setzen – und mit »ri« bestimmen, in welche Richtung gezielt werden soll – wird entweder ein nach links oder ein nach rechts geneigter Strich gezeichnet (Zeile 480 und 490). Mit Zeile 500 beginnt dann die Spielsteuerung. Drei Versuche sind vorgegeben, weshalb die Schleife in Zeile 500 von 1 bis 3 läuft. Vor jedem Spiel muß dabei natürlich der Punktestand 0 betragen (pu=0). In Zeile 510 und 520 wird nun dieser Stand und die Nummer des aktuellen Schusses (1, 2 oder 3) ausgedruckt. Anschließend werden zwei Unterprogramme aufgerufen und dann der nächste Versuch durch das NEXT in Zeile 540 gestartet. Es folgt die Endauswertung, wobei in Abhängigkeit von den erreichten Punkten ein passender Kommentar ausgegeben wird. Diese Art der Ausgabe kennen Sie ja bereits aus der Einleitung. Wir brauchen sie daher nicht weiter zu besprechen. Wichtig ist nur, daß nach der Ausgabe jeden Kommentares mit Hilfe des GOTO 630 zur Zeile 630 verzweigt wird, wo der letzte Punktestand ausgegeben wird. Mit einer Endabfrage wird zum Schluß geprüft, ob ein weiteres Spiel gewünscht wird oder der ganzen Sache ein Ende bereitet werden soll.

Die Eingabeauswertung finden Sie dabei in Zeile 650. Mit INKEY$ wird ein Zeichen in die Variable z$ eingelesen. Die LOWER$-Funktion ist dabei von größtem Nutzen. Sie wandelt einen Buchstaben in den dazugehörenden Kleinbuchstaben (beispielsweise »B« in »b«). Wie Sie vielleicht wissen, unterscheidet der Schneider bei der Abfrage zwischen Klein- und Großbuchstaben. Wenn Sie also statt des geforderten kleinen »j« den zugehörigen Großbuchstaben eingeben, erkennt der CPC das »j« nicht und bleibt in der Abfrageschleife hängen. Denn der Computer sucht ein kleines »n« – in diesem Fall beendet er das Programm durch END-oder ein kleines »j« – was eine Rückkehr zur Zeile 230 bewirkt. Alle anderen Eingaben, also auch die entsprechenden Großbuchstaben, führen nur wieder zum Zeilenanfang – zu einer erneuten Abfrage, zurück. Durch LOWER$ werden alle Großbuchstaben in Kleinbuchstaben umgewandelt und damit auch die Großbuchstaben für die Eingabe akzeptiert.

Nun werden Sie sich sicher fragen, wo die Steuerung des Balles und des Abschußgerätes geblieben ist. Dazu gehen wir einige Zeilen zurück, bis zum Aufruf der beiden Unterprogramme in Zeile 530. Hier findet die Steuerung für Ball und Armbrust statt. Vor dem Abschuß des Balles muß zunächst die »Rampe« gesetzt und gerichtet werden. Die dazu notwendigen Funktionen finden Sie in dem Unterprogramm ab Zeile 690.

Wir haben das Abschußgerät in den Zeilen 470 bis 490 bereits erstmalig dargestellt. Es soll nun aber möglich sein, vor jedem Schuß die Armbrust auf einen neuen Punkt zu setzen. Daneben soll auch zwischen Schießen nach rechts und nach links unterschieden werden. Wie dies zu machen ist, das sehen Sie ab Zeile 690.

Hier sind allerdings zunächst einige Hintergrundinformationen zur Tastaturabfrage notwendig. Wenn wir mit INKEY$ ein Zeichen vor der Tastatur einlesen, beispielsweise ein »n«, so enthält der Zeichenstring z$ nachher dieses Symbol. Mit »ASC(z$)« können wir dann den zugehörigen Code feststellen. Bei »n« wäre das 110. Wenn Sie nun im Anhang Ihres Bedienerhandbuchs nachschlagen, sehen Sie, daß das Zeichen 110 die Zeichenmatrix für das »n« enthält. Mit »PRINT CHR$(110)« wird deshalb auch ein kleines »n« dargestellt. Nun sind aber nicht nur den Tasten für Zahlen, Buchstaben und Sonderzeichen solche Zeichen zugeordnet. Auch die Cusorsteuertasten machen hier keine Ausnahme. Die Cursor-rechts-Taste liefert bei der INKEY$-Abfrage einen Pfeil nach links (Zeichen Nummer 242). Bei Cursor-links ist es ein Pfeil in der umgekehrten Richtung. Die COPY-Taste ist mit dem Zeichencode 224 verbunden. Wenn wir also z$ mit INKEY$ laden und dann mit ASC den Zeichencode feststellen, so können wir überprüfen, ob und wenn ja, welche Cursortaste gedrückt wurde.

Bewegung ins Spiel

Bei z=ASC(z$)(=242) (Cursor links) wird xfn – beim Aufruf der Routine ist dieser Wert gleich der X-Koordinate der Armbrust – um eins vermindert, dann mit dem ersten LOCATE in Zeile 760 die alte Stellung der Abschußrampe gelöscht, Ihre X-Koordinate auf den neuen Wert gesetzt (Programmzeile 770) und danach die Rampe an der neuen Position dargestellt (Zeile 790 oder 800). Analog läuft die Ausgabe bei Cursor-rechts ab. Hier wird die X-Koordinate um eins erhöht, dann wieder die alte Position gelöscht und die Rampe an der neuen dargestellt.

Wenn wir gleichzeitig mit den beiden Cursortasten SHIFT drücken, erhöhen sich die Zeichencodes um 4. Aus 242 wird damit 246 und in diesem Fall wird die Richtungsvariable ri umdefiniert (Zeile 740 und 750). Da danach die

Fortsetzung auf Seite 32



Spitzen-Software für Schneider-Computer und Commodore 128 PC

BRANDNEU
Jetzt auch für den Schneider Joyce

WordStar 3.0 mit MailMerge Der Bestseller unter den Textverarbeitungsprogrammen für PCs bietet Ihnen bildschirmorientierte Formatierung, deutschen Zeichensatz und DIN-Tastatur sowie integrierte Hilfstexte. Mit MailMerge können Sie Serienbriefe mit persönlicher Anrede an eine beliebige Anzahl von Adressen schreiben und auch die Adreßaufkleber drucken.
WordStar/MailMerge für den Schneider CPC 464*, CPC 664*
Bestell-Nr. MS 101 (3"-Diskette)
Bestell-Nr. MS 102 (5¼"-Diskette im VORTEX-Format)
WordStar/MailMerge für den Schneider CPC 6128
Bestell-Nr. MS 104 (3"-Diskette)
WordStar/MailMerge für den Schneider Joyce PCW 8256
Best.-Nr. MS 105 (3"-Diskette)
Hardware-Anforderungen: Schneider CPC 464*, CPC 664*, CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle
* Der Standard-Speicherplatz beim CPC 464/664 erlaubt ohne Speichererweiterung Blockverschiebe-Operationen nur bedingt und Simultan-Drucken gar nicht.
WordStar/MailMerge für den Commodore 128 PC
Bestell-Nr. MS 103 (5¼"-Diskette)
Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Markt&Technik
Schneider CPC-Software



mit MailMerge für den Schneider CPC 464/664

3" Schneider-Format

Und dazu die weiterführende Literatur:

WordStar für den Schneider CPC
Best.-Nr. MT 779, ISBN 3-89090-180-8
WordStar für den Commodore 128 PC
Best.-Nr. MT 780, ISBN 3-89090-181-6



dBASE II, Version 2.41 dBASE II, das meistverkaufte Programm unter den Datenbanksystemen, eröffnet Ihnen optimale Möglichkeiten der Daten- u. Dateihandhabung. Einfach u. schnell können Datenstrukturen definiert, benutzt und geändert werden. Der Datenzugriff erfolgt sequentiell oder nach frei wählbaren Kriterien, die integrierte Kommandosprache ermöglicht den Aufbau kompletter Anwendungen wie Finanzbuchhaltung, Lagerverwaltung, Betriebsabrechnung usw.
dBASE II für den Schneider CPC 464*, CPC 664*
Bestell-Nr. MS 301 (3"-Diskette)
Bestell-Nr. MS 302 (5¼"-Diskette im VORTEX-Format)
dBASE II für den Schneider CPC 6128
Bestell-Nr. MS 304 (3"-Diskette)
dBASE II für den Schneider Joyce PCW 8256
Best.-Nr. MS 305 (3"-Diskette)
Hardware-Anforderungen: Schneider CPC 464*, CPC 664*, CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle
* dBASE II für den Schneider CPC 464/664 ist lauffähig mit der VORTEX-Speichererweiterung auf 128 KByte. Diese erhalten Sie direkt bei der Firma VORTEX oder bei Ihrem Computerhändler.
dBASE II für den Commodore 128 PC
Bestell-Nr. MS 303 (5¼"-Diskette)
Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Markt&Technik
Schneider CPC-Software
dBASE™ II
ASHTON-TATE
für den Schneider CPC 6128
3" Schneider-Format

dBASE II für den Schneider CPC
Best.-Nr. MT 837, ISBN 3-89090-188-3
dBASE II für den Commodore 128 PC
Best.-Nr. MT 838, ISBN 3-89090-189-1



MULTIPLAN, Version 1.06 Wenn Sie die zeitraubende manuelle Verwaltung tabellarischer Aufstellungen mit Bleistift, Radiergummi und Rechenmaschine satt haben, dann ist MULTIPLAN, das System zur Bearbeitung »elektronischer Datenblätter«, genau das richtige für Sie! Das benutzerfreundliche und leistungsfähige Tabellenkalkulationsprogramm kann bei allen Analyse- und Planungsberechnungen eingesetzt werden wie z. B. Budgetplanungen, Produktkalkulationen, Personalkosten usw. Spezielle Formatierungs-, Aufbereitungs- und Druckanweisungen ermöglichen außerdem optimal aufbereitete Präsentationsunterlagen!
MULTIPLAN für den Schneider CPC 464*, CPC 664*
Bestell-Nr. MS 201 (3"-Diskette)
Bestell-Nr. MS 202 (5¼"-Diskette im VORTEX-Format)
MULTIPLAN für den Schneider CPC 6128
Bestell-Nr. MS 204 (3"-Diskette)
MULTIPLAN für den Schneider Joyce PCW 8256
Best.-Nr. MS 205 ("-Diskette)
Hardware-Anforderungen: Schneider CPC 464*, CPC 664*, CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle
* MULTIPLAN für den Schneider CPC 464/664 ist lauffähig mit der VORTEX-Speichererweiterung auf 128 KByte.
MULTIPLAN für den Commodore 128 PC
Bestell-Nr. MS 203 (5¼"-Diskette)
Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Markt&Technik
128er-Software



für den Commodore 128 PC
5¼"-Diskette im Floppy 1541-Format

MULTIPLAN für den Schneider CPC
Best.-Nr. MT 835, ISBN 3-89090-186-7
MULTIPLAN für den Commodore 128 PC
Best.-Nr. MT 836, ISBN 3-89090-187-5

Jedes Buch kostet DM 49,–
(sFr. 45,10/öS 382,20).
Erhältlich bei Ihrem Buchhändler.



Sie erhalten jedes **WordStar-, dBASE II-** und **MULTIPLAN-Programm** für Ihren Schneider-Computer oder Commodore 128 PC fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht. Die VORTEX-Speichererweiterung für den Schneider CPC 464 erhalten Sie direkt bei der Firma VORTEX oder bei Ihrem Computerhändler.

Diese Markt & Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser, bei Ihrem Computerhändler oder im Buchhandel.

Wenn Sie direkt beim Verlag bestellen wollen: gegen Vorauskasse durch Verrechnungsscheck oder mit der eingehefteten Zahlkarte.

Bestellungen im Ausland bitte an untenstehende Adressen.
Schweiz: Markt & Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, ☎ 042/41 56 56
Österreich: Überreuter Media, Handels- und Verlagsges. mbH, Alser-Str. 24, A-1091 Wien, ☎ 02 22/48 15 38-0
Für Auskünfte steht Ihnen Herr Teller, Telefon 0 89/46 13-205, gerne zur Verfügung.

Jedes Programm kostet DM 199,–*

(sFr. 178,–/öS 1890,–*)

* inkl. MwSt. Unverbindliche Preisempfehlung



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Ausgabe-Routine der Armbrust steht und diese von ri abhängt (Zeile 790 und 800), wird auch die Richtungsänderung sofort auf dem Bildschirm sichtbar. Nach dem Setzen geht es zurück zur Abfrage in Zeile 690. Hier wird festgestellt, ob die Schußrichtung der Armbrust (oder ihre Position) verändert werden sollen. Dann wird wieder das Abschußgerät ausgegeben. Der einzige Ausweg aus dieser Endlosschleife ist die COPY-Taste. Sie ist mit dem Zeichencode 224 belegt. Wenn Sie also auf COPY drücken, wird z=ASC(z$) (=224) gesetzt und dann kehrt das Programm (Zeile 730) zurück in die Hauptroutine.

Der zweite Befehl in Zeile 530 ruft aber schon wieder ein Unterprogramm (diesmal Ball) auf. Das Drücken der COPY-Taste ist ja für den Computer das Signal, daß der Ball freigegeben ist (Abschußposition und -richtung liegen vor). Der nun folgende Programmteil stellt im wesentlichen eine Simulation dar. Der Schneider simuliert den Flug des Balles anhand der Reflexionsgesetze und überprüft in der Routine, ob der Ball auf ein Herzchen oder ein Hindernis trifft.

Das sieht auf den ersten Blick sehr schwierig aus – ist es aber nicht. Schauen wir uns an, was wir bereits zur Verfügung haben. Ausgangspunkt der Flugbewegung ist die Abschußrampe mit den Koordinaten xf, yf. Auf diese Position setzen wir den Ball (Koordinaten x und y in Zeile 850). Nun überlegen wir uns, in welche Richtung der Ball abgestoßen werden soll. In jedem Fall soll er von unten nach oben fliegen. Da der Schneider die Zeilen von oben nach unten zählt (Zeile 1 ist also die oberste), müssen wir – um von einer Zeile zur nächsthöherliegenden zu kommen – von der Zeilenangabe, der Koordinate y, eins abziehen. Die Bewegung in Y-Richtung wird daher in Programmzeile 860 durch -1 ersetzt. Nun kann der Ball aber sowohl nach rechts als auch nach links fliegen. Die Flugrichtung ist in der Variablen ri festgelegt. Diese kennen Sie schon aus dem ersten Unterprogramm. Sie legt die Richtung unserer Abschußrampe fest. Bei ri=-1 ist sie nach links geneigt, ansonsten nach rechts. ri gibt damit also auch den Wert

```
10  '********************                  [81E0]
20  '** Monsterfressen **                  [0EDE]
30  '**                **                  [7AA4]
40  '**       von      **                  [908C]
50  '**                **                  [E4A8]
60  '** Carsten Straush**                  [1A9E]
70  '********************                  [EFEC]
80  '**********************                [7442]
90  '** Einleitungstext **                 [4FC8]
100 '***********************               [1494]
110 MODE 1                                 [8A50]
120 PRINT                                  [5F80]
130 PRINT"Sie sind einem boesen Monster
    in die groben Pfoten gefallen. Entwe
    der Sie besor-gen dem Kleinen etwas
    zu fressen, oder{2 SPACE}es wird mit
     Ihnen Vorlieb nehmen!"               [0D72]
140 PRINT:PRINT"An der Decke haengen meh
    rere Esswaren,{2 SPACE}die Sie mit e
    inem Ball treffen koennen.            [ADDE]
150 PRINT:PRINT"Den Ball schiessen Sie m
    it COPY ab. Mit "+CHR$(242)+" und "+
    CHR$(243)+" wird die Abschusslanze{2
     SPACE}bewegt. Bei gleichzeitigem Dr
    uecken von{2 SPACE}Shift{2 SPACE}die
     Richtung geaendert."                 [2EDE]
160 PRINT:PRINT"Doch denken Sie daran: D
    er{2 SPACE}Hunger{2 SPACE}des Tierch
    ens ist grossssssss!!"                [E3CE]
170 PRINT                                  [428A]
180 PRINT"Anfang mit <Space>"              [388A]
190 z$=INKEY$:IF z$<>" " THEN 190          [6A4E]
200 '************************              [28EA]
210 '** Anfangsparameter **                [3682]
220 '************************              [5AEE]
230 CLEAR:DEFINT a-z                       [A470]
240 INK 0,0:INK 1,24:INK 2,10:INK 3,6:BO
    RDER 10                                [7C10]
250 PAPER 0:PEN 1                          [7CDA]
260 WINDOW#0,1,35,1,25:WINDOW#1,36,40,1,
    25                                     [C07E]
270 PAPER#1,2:PEN#1,1                      [EC62]
280 CLS:CLS#1                              [F418]
290 '*******************                   [08AC]
300 '** Vorbereitung **                    [3F7C]
310 '*******************                   [A69E]
320 DIM o(40,25)                           [F3CC]
330 ri=1                                   [1CFE]
340 xf=20:yf=25:x=xf:y=yf:xfn=xf           [D75C]
350 FOR i=1 TO 40                          [54BA]
360 a=RND(1)*33+2:b=RND(1)*21+2            [60A6]
370 o(a,b)=2                               [92B0]
380 LOCATE a,b:PRINT CHR$(232)             [3EA4]
390 NEXT                                   [83F6]
400 FOR i=1 TO 60                          [5DB6]
410 a=RND(1)*33+2:b=RND(1)*21+2            [559E]
420 o(a,b)=1                               [93A6]
430 LOCATE a,b:PRINT CHR$(228)             [E3A6]
440 NEXT                                   [66EE]
450 LOCATE#1,1,3:PRINT#1,"Pkte."           [AF9C]
460 LOCATE#1,1,6:PRINT#1,"Schu"+CHR$(177
    )                                      [C67E]
470 LOCATE x,yf                            [612C]
480 IF ri=-1 THEN PRINT CHR$(205)          [02CC]
490 IF ri=1 THEN PRINT CHR$(204)           [A472]
500 pu=0:FOR j=1 TO 3                      [896C]
510 LOCATE#1,2,7:PRINT#1,j                 [B868]
520 LOCATE#1,2,4:PRINT#1,pu                [3B5A]
530 GOSUB 690:GOSUB 850                    [6DDC]
540 NEXT j                                 [4004]
550 '********************                  [C6FE]
560 '** Endauswertung **                   [506C]
570 '********************                  [FF02]
580 MODE 1                                 [2666]
590 IF pu<15 THEN PRINT"Sie sind so unfa
    ehig, dass es mir{2 SPACE}rich-tigen
     Spass machen wird, Sie zu fertil- g
    en!":GOTO 630                          [401E]
600 IF pu<25 THEN PRINT"Satt{2 SPACE}wer
    den{2 SPACE}kann man davon aber nich
    t!Sie haben Glueck, dass ich heute v
    ergetarisch eingestellt bin.":GOTO 6
    30                                     [36EA]
610 IF pu<35 THEN PRINT"Na ja. Fuer eine
    n kleinen{2 SPACE}Imbiss{2 SPACE}wir
    d es reichen!":GOTO 630                [FC7C]
620 PRINT"Schon ganz gut, fuer's erste!"
                                           [6D80]
630 PRINT:PRINT"Ihre Punktzahl";pu         [DCB4]
640 PRINT:PRINT"Noch ein Spiel j/n"        [E6F4]
650 z$=LOWER$(INKEY$):IF z$="n" THEN END
     ELSE IF z$<>"j" THEN 650 ELSE 230     [91B4]
660 '****************                      [D006]
670 '** SUB:Rakete **                      [2698]
680 '****************                      [220A]
690 z$=INKEY$:IF z$="" THEN 690            [DDA8]
700 z=ASC(z$)                              [6768]
710 IF z=242 THEN xfn=MAX(2,xf-1)          [B6FE]
720 IF z=243 THEN xfn=MIN(34,xf+1)         [8764]
730 IF z=224 THEN RETURN                   [820E]
740 IF z=246 THEN ri=-1                    [5C44]
750 IF z=247 THEN ri=1                     [94EE]
760 LOCATE xf,yf:PRINT" "                  [3C52]
770 xf=xfn                                 [624A]
780 LOCATE xf,yf                           [D900]
790 IF ri=-1 THEN PRINT CHR$(205)          [03D4]
800 IF ri=1 THEN PRINT CHR$(204)           [4068]
810 GOTO 690                               [E662]
820 '***************                       [9C5A]
830 '** SUB:Ball **                        [C5D2]
840 '***************                       [FA5E]
850 x=xf:y=yf:z=0                          [D280]
860 yr=-1:xr=ri                            [EC00]
870 x=x+xr:y=y+yr                          [EE00]
880 LOCATE x,y:PRINT" ";                   [0C36]
890 x=x+xr:y=y+yr                          [B404]
900 '****************************          [8DF0]
910 '** CHECK:Bande&Hinderniss **          [CBF0]
920 '****************************          [07F4]
930 f=0                                    [3D1E]
940 IF o(x,y)<>2 THEN 990                  [F90C]
950 xr=SGN(RND(1)-0.5):IF xr=0 THEN 950    [9122]
960 yr=SGN(RND(1)-0.5):IF yr=0 THEN 960    [C92A]
970 GOTO 890                               [0574]
980 yr=SGN(RND(1)-0.5):IF yr=0 THEN 980    [BD32]
990 IF x<1 THEN xr=-xr:f=1                 [0E22]
1000 IF y<1 THEN yr=-yr:f=1                [5166]
1010 IF x>35 THEN xr=-xr:f=1               [08D4]
1020 IF y>24 THEN yr=-yr:f=1               [EBD8]
1030 IF f=1 THEN 890                       [88EE]
1040 '**********************               [81FC]
1050 '** Ball darstellen **                [E57C]
1060 '**********************               [E200]
1070 LOCATE x,y:PEN 3:PRINT CHR$(227);     [3CB8]
1080 IF o(x,y)=1 THEN o(x,y)=0:pu=pu+1:L
     OCATE#1,1,4:PRINT#1,pu                [02D0]
1090 FOR t=1 TO 150:NEXT t                 [B4B2]
1100 z=z+1:IF z=100 THEN LOCATE x,y:PRIN
     T " ";:PEN 1:RETURN                   [2524]
1110 GOTO 880                              [96B8]
```

Listing. Achtung! »Monsterfressen« ist nicht nur zum Spielen gedacht. Lernen Sie programmieren unter Basic

wieder, um den wir die X-Koordinate bei jedem Schritt verändern müssen. Soll der Ball nach links fliegen, so muß die X-Verschiebung xr=-1 gesetzt sein, und umgekehrt xr=1 bei einer nach rechts gerichteten Flugbahn. Um von einem Punkt auf der Flugbahn des Balles den nächsten zu berechnen, müssen wir nur die aktuelle X- beziehungsweise Y-Verschiebung addieren. Dies macht Zeile 870. Ein Beispiel: Wenn die Abschußrampe die Koordinaten (20, 25) besitzt und die Abschußrichtung »rechts« (ri=1) ist, dann ergibt sich als erste Position des Balles (x+xr,y+yr)=(20+1,25-1)=(21,24) also der Bildschirmpunkt rechts oberhalb der Armbrust. Spielen Sie diese Setzbewegung einmal in Gedanken für die andere Abschußrichtung durch.

Wir lassen im Moment die Zeilen 900 bis 1030 unberücksichtigt und betrachten zu Anfang den Rest des Unterprogramms. Das Setzen des Balles geschieht in drei Schritten. Zunächst muß an der alten Position der noch von der letzten Bewegung her gesetzte Ball gelöscht werden. Dies geschieht wie bei unserer Abschußrampe durch die Ausgabe eines positionsrichtigen Leerzeichens. Vergleichen Sie dazu einmal die Programmzeilen 880 mit 760. Danach wird die neue Position, wie gerade schon beschrieben, errechnet. Wenn Sie nun den nächsten Teil übergeben, sehen Sie ab Zeile 1070 die Ausgabe des Balls. Als Symbol wird CHR$(227) verwendet. Das Prinzip kennen sie ja schon. Mit LOCATE wird der Cursor auf die neue Ballposition gesetzt und dann mit einem PRINT-Befehl das Ballsymbol ausgegeben. Damit sich der Ball auch farblich deutlich abhebt, wird vor der Ausgabe des Zeichens dabei mit PEN die Schriftfarbe umgeschaltet. Zeile 1080 überprüft nun, ob an der aktuellen Ballposition ein Freßsymbol gespeichert war. Wenn dies der Fall (o(x,y)=1) ist, so wird der Punktezähler pu um 1 erhöht und der neue Punktestand im Window #1 ausgegeben. Damit ein zweiter Durchlauf das Herzchen nicht noch einmal findet, muß das entsprechende Feld im Array o(x,y) gelöscht werden. Es wird wie alle anderen leeren Felder mit 0 geladen. Zeile 1090 beinhaltet eine einfache Zeitschleife. Der Computer zählt dabei Schäfchen von 1 bis 150. Einziger Sinn dieser Operation: Zeit verprassen. Während wir Menschen nämlich öfters in Zeitprobleme kommen, ist der Schneider bei einfachen Anwendungen wie hier so schnell, daß wir gar nichts mehr erkennen können. Der Ball würde nur so über das Bild flitzen. Wenn Sie einmal sehen wollen, wie schnell der Schneider unser Programm bearbeiten kann, brauchen Sie nur diese Zeile weglassen. Die Zeitverzögerung unterbleibt dann.

Nun wäre es relativ sinnlos, den Ball ewig über den Schirm sausen zu lassen. Normalerweise besitzt ein geworfener Gegenstand ja nur eine bestimmte Energie. Nach deren Verbrauch fällt er auf die Erde zurück. Bei unserem Programm wird dazu die Variable z benutzt. Beim Aufrufen des Unterprogramms ab Zeile 850 wird z=0 gesetzt. Mit jedem Schritt, den der Ball nun auf dem Bildschirm macht, wird z in Zeile 1100 um eins erhöht. Ist z kleiner als 100, kehrt die Routine aufgrund des GOTO-Befehls in 1110 wieder an den Anfang (Zeile 880) zurück. Hier wird die alte Ballposition gelöscht und die neue berechnet. Danach wird die Darstellung des Balls wiederholt. Bei z=100 wird der Ball durch nochmaliges LOCATE und Ausgabe eines Leerzeichens zwar noch gelöscht, aber dann kehrt der Computer ins Hauptprogramm (Zeile 540) zurück.

Der »hüpfende« Ball

Soweit zur Erklärung der eigentlichen Flugsimulationsroutine. Nun müssen bei der Bewegung des Balles allerdings noch zwei weitere Punkte beachtet werden. Der letzte Teil des Unterprogramms Ball (zwischen Zeile 930 und 1030) beschäftigt sich damit. Erstens kann der Ball auf die Randbegrenzung treffen. In diesem Fall muß er nach den Reflexionsgesetzen zurückgeworfen werden. Im zweiten Fall, wenn der Ball gegen ein Hindernis prallt, soll eine zufällige Ablenkung erfolgen.

Die Hindernis-Behandlung geschieht in den Zeilen 940 bis 970. Zuerst wird gefragt, ob an der neuen Position des Balles überhaupt ein Hindernis steht. Ist o(x,y) < > 2, so ist an der aktuellen Ballposition kein Hindernis vorhanden und das Programm verzweigt aus Zeile 940 nach 990. Ansonsten muß die Richtung geändert werden. Dies ist relativ einfach zu bewerkstelligen. Wir müssen nur mit der Zufallsfunktion RND() zufällige Werte für die Verschiebungen xr und yr berechnen. Dies erfolgt in Zeile 950 beziehungsweise 960. Dabei muß man allerdings beachten, daß keiner der beiden Parameter 0 werden darf, da ja ansonsten keine Verschiebung mehr stattfinden würde. Tritt also der Nullfall ein, so probiert es der Computer einfach noch einmal, indem er wieder zum Anfang der Zeile zurückkehrt. Nachdem die neuen Veränderungsparameter erzeugt wurden, muß nun noch die aktuelle Position des Balles mit deren Hilfe errechnet werden. Durch die Rückkehr aus Zeile 970 nach 890 wird dies gemacht. Eine neue Position wird bestimmt, und dann gegebenenfalls durch den Teil ab Zeile 1070 der Ball an dieser Stelle auf dem Schirm dargestellt. Es kann jedoch sein, daß sich an der durch Zufall bestimmten neuen Position wiederum ein Hindernis befindet. Dann wird noch einmal durch den Teil ab Zeile 940 eine Zufallsposition bestimmt. Dieses Spielchen läuft solange ab, bis eine gültige Position erreicht wird und der Ball gesetzt werden kann.

Auch der zweite Fall, die Bandenprüfung und dementsprechende Korrektur, erfolgt analog. In Zeile 990 wird überprüft, ob sich die Position, an der der Ball neu gesetzt werden soll, außerhalb des Bildschirms befindet und zwar in Zeile 990 links vom linken Bildschirmrand. In diesem Fall wird das Vorzeichen des Verschiebeparameters xr umgekehrt. Der Ball fliegt damit rückwärts in bezug auf seine vorherige Position (also wieder nach rechts). Nach demselben Prinzip laufen die Korrekturen für die Randbegrenzungen in den anderen drei Richtungen ab. Zeile 1000 überprüft, ob der Ball bei einem neuen Setzen den Schirm nach oben verlassen würde (y<1). Zeile 1010 korrigiert die Flugbahn bei Überschreiten des rechten Randes und 1020 bei Ausbruch nach unten. Dabei ist zu berücksichtigen, daß die verschiedenen Vergleiche nacheinander und unabhängig voneinander ablaufen. Wenn der Ball über die linke obere Ecke von rechts-unten kommend den Schirm verlassen will, so wird gleichzeitig x<1 und y<1. Entsprechend werden beide Richtungsparameter (xr und yr) umgekehrt. Bei jeder Änderung wird dabei der beim Aufruf dieses Programmteils (Zeile 930) rückgesetzte Merker f gesetzt. Zeile 1030 verweist den CPC dann auch bei gesetztem f nach Zeile 890 zurück, wo wieder die nächste Position bestimmt wird. Auch hierauf geht es durch die Abfolge der Zeilen immer erst einmal zurück in den Check-Teil ab Zeile 910. Hindernisse und Rand werden geprüft, bevor dann – gegebenenfalls auch erst nach 10 oder mehr vergeblichen Versuchen und Reflexionen – eine Position erreicht wird, die allen Anforderungen genügt. Erst dann kann die nächste Ballposition berechnet werden.

Wenn Sie das Programm bis jetzt noch nicht fertig eingetippt haben, holen Sie dies nach und lassen Sie es laufen. Mit den bisherigen Informationen für Sie ist es kein Problem mehr, das Programm zum Laufen zu bringen. Trotz der vielen Erklärungen dürfen Sie nicht vergessen, daß Computern auch Spaß und nicht nur Arbeit bedeutet. Gönnen Sie sich ein paar Spiele mit unserem Programm. Viel Spaß dabei.

(Carsten Straush/hg)

Gekonnter Bildschirmaufbau



Eine gute Bildschirmausgabe ist das I-Tüpfelchen eines Programms. Das Schneider-Basic hält dafür einen leistungsstarken Befehl bereit: PRINT USING

Ein chinesisches Sprichwort lautet: »Ein Bild sagt mehr als tausend Worte«. Analog dazu könnte man bei der Computerei sagen: Ein Programm ist so gut oder schlecht wie sein Bildschirmaufbau. Auch ein noch so gutes Programm wird abqualifiziert, wenn man mit den von ihm gelieferten Informationen nur schwer etwas anfangen kann, weil ein ungünstiges Ausgabeformat gewählt wurde. Ein einfaches Beispiel mag dies illustrieren. Nehmen wir einmal an, wir wollen drei Vergleichswerte – beispielsweise die Telefonkosten dreier aufeinanderfolgender Monate – im Rahmen einer Haushaltskostenberechnung auf dem Bildschirm anzeigen. Betrachten Sie nun die beiden folgenden Darstellungen:

```
129.5     89.64     76
```

und

```
Telefonkosten
April DM 129.50
Mai   DM  89.64
Juni  DM  76.00
```

Es ist sicher keine Frage, welche Ausgabe besser lesbarer ist. Zugegebenerweise wurde hier ein besonders krasser Gegensatz gewählt. Dies ändert jedoch nichts an der prinzipiellen Forderung, den Bildschirm so zu gestalten, daß beispielsweise Vergleichswerte auch vergleichbar untereinander angeordnet sind – und zwar stellenrichtig. Die Wertigkeiten der untereinanderstehenden Stellen dürfen also nicht verschieden sein, da sonst ein falscher Eindruck entsteht. Es reicht somit nicht aus, wenn wir die drei Zahlen einfach untereinander drucken.

```
129.5
89.64
76
```

Im Vergleich zur zweiten Darstellungsart ist die Übersichtlichkeit dieser dritten immer noch mangelhaft. Was fehlt, ist ein geordnetes Format. Um das Ganze leichter zu verstehen, finden Sie hier zuerst einmal eine Definition:

Unter formatierter Ausgabe versteht man die Darstellung eines Textes oder einer Variablen in einem vorgegebenen Format. Dabei wird die betreffende Größe (Text oder numerische Variable) mit Hilfe von Formatierungszeichen und Platzhaltern auf ein bestimmtes Format »getrimmt«. Somit wird der wirklich wichtige Teil eines Textes oder einer numerischen Variablen herausgehoben. Ein kurzes Beispiel verdeutlicht dies.

In einer Steuerberechnung, in der vom Verkaufspreis die Mehrwertsteuer abgezogen werden soll, ist es naturgemäß vollkommen uninteressant, welchen Wert die dritte oder vierte Nachkommastelle annimmt. Die Ausgabe eines hochpräzisen Zahlenbandwurms würde die Lesbarkeit der Berechnung nur erschweren, ohne zusätzliche Informationen zu bringen. Bei einer Tabellendarstellung kann das sogar dazu führen, daß die letzten Ziffern bereits in die nächste Spalte reichen und dort befindliche Werte überschreiben. Weniger ist hier also mehr. Nun werden aber beim Schneider Zahlen, soweit nichts anderes angegeben wurde, mit bis zu sieben Nachkommastellen ausgegeben. Das Problem hier also zunächst einmal darin, die Werte auf die entsprechende Länge zu verkürzen.

Dies könnten wir bei numerischen Variablen mit den beiden Funktionen ROUND und FIX erreichen. FIX schneidet alle Nachkommastellen ohne Rundung ab. ROUND rundet auf die angegebene Stellenzahl. Wenn wir beispielsweise unsere drei oben aufgeführten Werte mit »ROUND (<Wert>, 0)« ausgeben, so sind wir zumindest das Problem der unterschiedlichen Anzahl von Nachkommastellen los. Wir verändern dadurch allerdings den Wert der entsprechenden Variablen. Wollen wir aber die Genauigkeit erhöhen und daher mit »ROUND (<Wert>,2)« arbeiten, so gibt es schon den ersten Ärger. Ihr Computer unterschlägt nämlich alle Nachkommanullen. Wir haben es wieder mit unterschiedlich langen – und damit schlecht zu positionierenden – Ausgabegrößen zu tun.

Zur Lösung derartiger Probleme stellt das Locomotive-Basic des Schneiders das Kommando PRINT USING zur Verfügung. Dieser Befehl besteht aus einer einfachen PRINT-Anweisung mit einem angehängten USING-Teil. Diese nachgestellte Spezifikation erlaubt uns, ein (fast) beliebiges Ausgabeformat festzulegen. Was das heißt, sollen zwei Beispiele verdeutlichen. Geben wir »a=27.75:PRINT a« ein, so wird der Wert 27.75 mit einer Vorleerstelle am linken Bildschirmrand ausgedruckt. Geben wir dagegen a unter einer USING-Bedingung durch »a=27.75: PRINT USING "##.#";a« aus, so erscheint der Wert 27.8 linksbündig auf dem Bildschirm. Was ist geschehen? Unsere Variable ist beim Ausdruck auf eine Nachkommastelle gerundet worden. Die Leerstelle, die der Schneider immer bei der Umwandlung einer Zahl in einen String voranstellt (unter anderem bei der Bildschirmausgabe), wurde unterdrückt. Durch PRINT USING erhalten wir also eine formatierte Ausgabe. Die Variable selbst blieb jedoch unverändert, wie Sie mit »PRINT a« einfach überprüfen können. Die Änderung der Zahl erfolgt also nur für die Ausgabe. Beim nächsten Beispiel zeigt sich noch ein anderer Effekt des angehängten USING. Durch »a=27.75:b=7.395: PRINT USING "##   .#";a:PRINT USING "##   .#";b« erhalten wir die Werte 27.8 und 7.4 auf dem Bildschirm. Diese werden aber nicht linksbündig ausgegeben, sondern schön mit dem Dezimalpunkt untereinander. Genauso, wie wir es in einer Tabellendarstellung benötigen und oben gefordert haben. Wodurch wird nun das Ausgabeformat bestimmt? Es wird durch Sonderzeichen festgelegt.

Schauen wir uns dazu den PRINT USING-Befehl – beziehungsweise das angehängte USING – näher an. Die Anweisung besteht aus drei Teilen. Am Anfang steht der USING-Befehl, am Ende die Liste der unter diesem Kommando auszugebenden Variablen. Dazwischen befindet sich die Spezifikation des Ausgabeformates. Die hier eingegebenen Zeichen haben entweder Platzhalterfunktion oder sind Fixtexte, die bei jeder Ausgabe mit auf den Schirm gebracht werden. Die meisten Variationsmöglichkeiten ergeben sich dabei für numerische Variable. Ein paar Beispiele:

»a=27.75:PRINT USING "DM## .#";a« ergibt den Ausdruck »DM 27.8«. Vor der Ausgabe unseres Wertes werden also die in der USING-Klausel angegebenen Zeichen (»DM«) mit ausgegeben. Wir können diese natürlich auch nachstellen. Mit »a=27.75:PRINT USING "##.# DM";a« erhalten wir DM nachgestellt. Soweit zu den Fixtexten.

Bestimmte Zeichen haben jedoch auch eine Sonderfunktion, so das schon verwendete Doppelkreuz (»#«). Es stellt einen Platzhalter für eine Stelle der numerischen Variablen dar. Die Variablen in dem Befehl werden also in

diesem Format ausgegeben. Das »#« vor dem Dezimalpunkt symbolisiert dabei eine Vorkommastelle, das »#« nach dem Punkt macht dasselbe für die Nachkommastellen der Variablen. Mit der Spezifikation »##.#«" haben wir also für die Variablen eine Nachkomma- und zwei Vorkommastellen festgelegt.

Hat die Variable weniger Stellen als durch Platzhalter freigehalten wurden, so werden vor dem Komma Leerstellen eingefügt. Besitzt unsere Variable nach dem Komma mehr Stellen, als durch die USING-Klausel angegeben, so wird auf die letzte Stelle – hier also die erste Nachkommastelle – gerundet. Bei freien Nachkommastellen werden die restlichen Plätze mit Nullen gefüllt.

Nun kann es aber passieren, daß mehr Vorkommastellen belegt werden müßten, als spezifiziert wurden. Ist der auszugebende Wert größer als erlaubt, in unserem Beispiel eine Zahl mit drei Vorkommastellen, so wird das Überlaufzeichen »%« und der volle ungekürzte Variablenwert ausgegeben. Dadurch wird natürlich unsere Tabelle nicht mehr spaltenrichtig ausgegeben. Wir müssen also unbedingt darauf achten, eine genügend große Anzahl von Vorkommastellen mit dem Doppelkreuz zu reservieren.

Mit der Angabe der Ausgabestellen und etwaiger zusätzlicher Symbole sind die Fähigkeiten von PRINT USING bei weitem noch nicht erschöpft. Als Ergänzung können Vorzeichen definiert im PRINT USING-Kommando ebenfalls berücksichtigt. Die beiden Leerzeichen zwischen dem »M« und der »2« ergeben sich somit als Summe aus dem definierten Leerzeichen und der einen nicht benötigten Stelle des Ausgabefeldes (erstes »#«). Geben wir in diesem Format den Wert 7.8 aus, so befinden sich drei Leerzeichen zwischen Zahlenwert und DM-Kürzel.

Etwas anderes bewirkt das »–«. Ein vorgestelltes Minus hat den gleichen Effekt wie alle anderen Zeichen; es wird immer vor dem Zahlenwert angegeben. Setzen wir dagegen ein »–« hinter die Kreuze, so erscheint es nur bei negativen Zahlen. Bei positiven Werten wird es durch ein Leerzeichen ersetzt. Ein »+« an dieser Stelle bewirkt hingegen die permanente Ausgabe des Vorzeichens.

Wir kommen nun zu einigen Spezialitäten. Ein vorangestelltes »*« verursacht die Ausgabe desselben. Zwei Sternchen dagegen füllen die nicht benötigten Stellen auf. Dabei werden vorab zwei Sternchen ausgegeben.

»a=28.7:PRINT USING"**###";a« bringt uns die Ausgabe »*****29« auf den Schirm. Die fünf Sternchen setzen sich dabei aus zwei permanent ausgegebenen Sternchen und den drei freien Stellen zusammen. Da wir keine Nachkommastellen angegeben hatten, wurde a auf 29 aufgerundet. Das Doppelsternchen wird zum Beispiel beim Ausdruck von Bilanzen

Hochpfeilen nach den Doppelkreuzen. Der Computer gibt dann die Zahl mit einer Vorkommastelle und Exponentenangabe aus. In unserem Beispiel würde mit »a=1000000:PRINT USING "#↑↑↑↑";a« die Million als »1E+06« auf dem Bildschirm dargestellt.

PRINT USING beschränkt sich aber nicht auf die Behandlung numerischer Größen. Für Texte stehen zwei weitere Formatierungszeichen zur Verfügung: das Ausrufezeichen (»!«) und der Backslash (»\«). Nach »a$="Testzeichenkette" und PRINT USING"!";a$« erhalten wir linksbündig ein großes »T« auf dem Bildschirm. Wird der String mit zwei Formatierungszeichen (»!!«) unter PRINT USING ausgegeben, so erscheinen die ersten beiden Buchstaben. Viel interessanter ist jedoch der Backslash (»\«). Dieses Formatierungszeichen erlaubt, Zeichenketten (Strings) auf eine vorgeschriebene Länge zu kürzen – beziehungsweise eine Zeichenkette durch Anhängen von Leerstellen auf die gewünschte Länge zu bringen. Geben wir nun »PRINT USING "\   \";a$« aus, so erscheint das Halbwort »Test«. Es wurden also die ersten fünf Zeichen berücksichtigt. Auch hier müssen wir wieder die Formatierungszeichen als Platzhalter mitrechnen: zwei Stellen für die Schrägstriche plus drei dazwischenliegende Leerzeichen. Alle zusammen liefern uns dann die Gesamtzahl der möglichen Stellen. Mit dem Backslash steht uns also ein Hilfs-

```
Schnellhefter       Stueck    185
Hefte DIN A4        Stueck    231
Hefte DIN A5        Stueck     12
Ordner              Stueck     56
Ringbuchpapier      Stueck    123
Schreibbloecke      Stueck     34
Loeschpapier        Stueck     57
Bleistifte          Stueck     96
Kugelschreiber      Stueck     75
Radiergummi         Stueck     32
```

Bild 1. Eine kleine Lagerverwaltung mit PRINT USING

```
10 FOR i=1 TO 10                                    [2546]
20 INPUT "Artikelname":n$(i)                        [B4D4]
30 INPUT "Anzahl":s(i)                              [437A]
40 NEXT i                                           [6198]
50 CLS                                              [E1CE]
60 FOR i=1 TO 10                                    [2050]
70 PRINT USING "\{18 SPACE}\":n$(i)::PRI
   NT USING " Stueck #####":s(i)                   [4224]
80 NEXT i                                           [81A0]
```

Listing 1. Acht Zeilen reichen für eine Lagerverwaltung aus

werden. Ein »+« gibt dabei immer das Vorzeichen aus. Es erscheint ein »+«, falls der Zahlenwert positiv ist, ein »–« bei negativen Werten. Das »+« kann dabei vor oder nach den Platzhaltern stehen.

»a=27,8:PRINT USING "DM ###.##+";a« erzeugt beispielsweise den folgenden Ausdruck: »DM 27.80+«. Die nicht benötigten Felder werden durch Leerzeichen (Vorkomma) oder Nullen (Nachkomma) aufgefüllt. Das nachgestellte »+« bewirkt die Ausgabe eines »+«, da die Variable positiv ist (sonst »–«). Das Zeichen DM wird vor der Variablenausgabe auf den Schirm gebracht, und auch die Leerstelle zwischen »DM« und den Platzhaltern wird mit ausgegeben. Leerzeichen werden

und anderen kaufmännischen Urkunden benutzt, um ein nachträgliches Ändern der Werte zu verhindern.

Für die Arbeit mit großen Zahlen stellt uns das PRINT USING-Kommando zwei spezielle Symbole zur Verfügung. Mit dem »,« können wir nach drei Stellen ein Komma ausgeben, was das Abzählen längerer Zahlen vereinfacht. Setzen wir a auf eine Million und geben diesen Wert dann mit »PRINT USING "##,###,####"« aus, so erhalten wir als Ergebnis »1,000,000«. Die unbenutzte Stelle wurde wieder durch ein Leerzeichen aufgefüllt, die Kommas stellenrichtig eingefügt. Bei sehr großen Zahlen bietet sich die Exponentialschreibweise an. Diese erreichen wir beim Schneider durch Eingabe von vier

mittel zur Verfügung, mit dem wir neben Zahlen auch Zeichenketten formatiert ausgeben können.

Wir brauchen uns aber nicht auf eine PRINT USING-Anweisung pro Bildschirmzeile zu beschränken. Die einzelnen Kommandos können auch addiert werden. Die dabei entstehenden »Befehls-Bandwürmer« sehen zwar auf den ersten Blick etwas kompliziert aus, ermöglichen aber eine höchst komplexe, sehr einfache und schnelle Programmierung. Anhand eines Beispiels ist das leicht zu sehen. Im Rahmen eines Lagerverwaltungsprogramms sollen Artikelname und aktuelle Stückzahl ausgegeben werden. Der Name des Artikels ist dabei in der Variablen n$(i), die Stückzahl in s(i) gespeichert.

Wir müssen uns zuerst Gedanken über das Ausgabeformat machen. Wir nehmen dazu an, daß der Artikelname maximal 20 Zeichen umfassen soll. Für die Stückzahl reichen in unserem Beispiel sechs Stellen aus. Jetzt können wir an die Definition des Formates gehen. Linksbündig soll der Artikelname ausgegeben werden, daneben das Wort Stück und dann die aktuelle Stückzahl (Bild 1). Eine solche Formatdefinition bezeichnet man auch als Maske. In unserem Fall besteht diese Ausgabemaske also aus einem 20-Zeichen-Feld für den Namen, dem Wort Stück und dann einem weiteren 6-Stellen-Feld für die Stückzahl. Mit PRINT USING können wir nun eine solche Ausgabe höchst komfortabel programmieren:

»PRINT USING″\\″;n$(i);:PRINT USING″ Stueck# # # # # #″;s(i)«

Das erste USING legt die maximale Zeichenzahl für den Text fest. Es folgt mit »;« angehängt die auszugebende Variable. Das zweite Semikolon weist den Computer an, die Zeile noch nicht abzuschließen. Ohne dieses Semikolon würde das nächste PRINT-Kommando unter das erste gedruckt werden. So aber wird es nahtlos angefügt. Zunächst kommt aber das Wort »Stueck« auf den Bildschirm, bevor dann die Variable s(i), auf sechs Stellen formatiert, folgt. Da wir nach dem zweiten Kommando kein weiteres »;« mehr angehängt haben, wird die Zeile abgeschlossen. Das nächste PRINT arbeitet also eine Zeile tiefer.

Wenn Sie diese Zeile eingeben und bearbeiten lassen, erhalten Sie natürlich nun noch keine sinnvolle Ausgabe, denn die Variablen n$(i) und s(i) enthalten ja noch keine Werte. Das können wir aber mit wenigen Programmzeilen ändern.

```
10 FOR i=1 TO 10
20 INPUT"Artikelname";n$(i)
30 INPUT"Anzahl";s(i)
40 NEXT i
```

Mit Hilfe dieser Zeilen können Sie nun 10 Artikel eingeben. Fügen Sie nun noch

```
50 CLS
60 FOR i=1 TO 10
80 NEXT i
```

ein und geben dem Formatbefehl die Zeilennummer 70. Das gesamte Programm finden Sie in Listing 1. Im Rahmen einer Gesamtübersicht werden jetzt die gespeicherten Artikel untereinander formatiert ausgegeben. Durch die Verwendung einer komplizierten Format-Anweisung innerhalb einer Schleife (wie in unserem Beispiel), kann also bereits ein komplexer Bildschirmaufbau mit relativ wenigen Kommandos erreicht werden.

Bei dieser Art der Programmierung gibt es jedoch auch noch eine andere Variante, die im Handbuch des Schneiders nicht zu finden ist. Statt der direkten Angabe der Formatierung innerhalb von Anführungsstrichen können wir auch mit Formatierungsstrings arbeiten. Dazu belegen wir eine Stringvariable – beispielsweise s$ – mit dem Ausdruck, den wir normalerweise hinter dem USING in Anführungszeichen angeben würden. Wenn wir beispielsweise »s$=″\ \″« eingeben, so haben wir damit einen solchen Formatstring erzeugt. Durch seine Verwendung wird das PRINT USING-Kommando sehr vereinfacht. Wir können nun nämlich unseren Text mit »PRINT USING s$;a$« ebenfalls auf 20 Zeichen Länge formatiert ausgeben. Diese Technik hat darüber hinaus den Vorteil, daß das USING-Format an anderer Stelle im Programm noch einmal benutzt werden kann. Wenn Sie also bei einer späteren Ausgabe wieder auf die Beschränkung auf 20 Zeichen zurückkommen wollen, so ist dies kein Problem mehr. Mit »PRINT USING s$;neustring$« kann auch die neue Zeichenkette neustring$ problemlos formatiert ausgegeben werden. Speziell wenn man viele Ausgaben unter derselben Formatierungsanweisung benötigt, kann man durch diesen Trick Speicherplatz sparen. Das Programm wird übersichtlicher und läuft auch ein wenig schneller.

Die bis jetzt beschriebenen Eigenschaften des PRINT USING-Befehls können wir teilweise auch auf andere Art erreichen – beispielsweise mit LOCATE oder mit Hilfe von Windows. PRINT USING hat jedoch einen großen Vorteil: Es funktioniert gleichermaßen für die Bildschirmausgabe und die Druckeransprache. Nach dem PRINT-Teil können wir nämlich mit »#« und einer nachgestellten Nummer einen anderen Ausgabekanal als den Hauptbildschirm (Nummer 0) wählen. Geben Sie zum Beispiel »PRINT#8, USING″DM # # # #.# #″;s« ein, so wird der Inhalt der Variablen s als DM-Betrag im schon gewohnten Format an den Drucker ausgegeben.

Nachdem wir nun einen detaillierten Überblick über die Leistungsfähigkeit des PRINT USING-Befehls gewonnen haben, wollen wir uns einmal anhand eines praktischen Beispiels damit auseinandersetzen, wie wir dieses Kommando im Rahmen einer einfachen Listenverarbeitung benutzen können. Als Beispiel dient ein einfaches Haushaltsbuch. Das ist eine Bilanz im kleinen, eine Übersicht über die laufenden und außergewöhnlichen Ausgaben, die das tägliche Leben mit sich bringt. Der Sinn und Zweck solcher Aufzeichnungen liegt darin, sich einmal vor Augen zu führen, was man eigentlich wofür ausgegeben hat. Damit schafft man sich eine Kontrollmöglichkeit, die es erlaubt festzustellen, ob sich eine Ausgabe gelohnt hat. Früher diente dazu ein einfaches kleines Heftchen, in das die einzelnen Kosten eingetragen wurden. Das Programm »Haushaltsbuch« stellt nun eine elektronische, der Computer-Zeit angepaßte, Variante dar.

Hier wird nichts mehr geschrieben, sondern einfach laufend eingegeben. Wer einen Drucker besitzt, kann sich vom aktuellen Stand eine Hardcopy, einen Bildschirmausdruck, erstellen. Haushaltsbuch ist dabei keineswegs eine optimierte und mit allem Komfort ausgestattete Programmversion. Sie soll nur das Prinzip demonstrieren und darüber hinaus verschiedene Anwendungsmöglichkeiten für PRINT USING zeigen. Erweitern Sie die vorgegebene Version aber ruhig nach Belieben.

Die Routine Haushaltsbuch benötigt nur relativ wenig Variablen. Insgesamt können 20 verschiedene Ausgabeposten verwaltet werden, deren Titel in den Variablen n$(1) bis n$(20), beziehungsweise deren Wert in n(1) bis n(20) abgelegt ist. Alle Ausgabeposten erscheinen dabei in einer großen Liste auf dem Bildschirm, wobei zunächst

	Titel	Betrag		Summe	
1	Lebensmittel	DM	-133.98	DM	-133.98
2	Koerperpflege	DM	-12.65	DM	-146.63
3	Autokosten	DM	-265.00	DM	-411.63
4	Miete	DM	-890.00	DM	-1301.63
5	Rundfunk	DM	-16.90	DM	-1318.53
6	Versicherung	DM	-113.87	DM	-1432.40
7	Heizkosten	DM	-65.00	DM	-1497.40
8	Kleidung	DM	-135.00	DM	-1632.40
9	Geschenke	DM	-25.00	DM	-1657.40
10	Hundefutter	DM	-43.76	DM	-1701.16
11	Buerobedarf	DM	-9.80	DM	-1710.96
12	Schulgeld	DM	-40.00	DM	-1750.96
13	Buecher	DM	-9.80	DM	-1760.76
14	Wassergeld	DM	-20.00	DM	-1780.76
15	Nebenkosten	DM	0.00	DM	-1780.76
16	Kino	DM	-24.00	DM	-1804.76
17	Theater	DM	-45.00	DM	-1849.76
18	Reparaturen	DM	-123.00	DM	-1972.76
19	Muellabfuhr	DM	-23.00	DM	-1995.76
20	sonstiges	DM	2498.87	DM	503.11

Bild 2. Führen Sie Ihr Haushaltsbuch mit dem Computer

links der Titel des Ausgabeposten, dann der aktuelle Wert und danach in einer Staffelrechnung die Summe aller bisherigen Ausgaben dargestellt werden. Der Wert ganz rechts in der untersten Zeile zeigt damit die Summe aller Ausgaben (siehe Bild 2). Nachdem am Anfang des Programms die gerade beschriebenen zwei Variablen dimensioniert wurden, werden danach die Namen der einzelnen Ausgabeposten definiert. Die ersten drei Definitionen sind dabei vorgegeben. Die Angaben für die Ausgabegruppen 4 bis 20 sind Ihnen überlassen. Sie sind allerdings natürlich auch bei den ersten drei Werten nicht an die Vorwahl gebunden. Die anderen Titel definieren Sie dabei wie in den Zeilen 50 bis 70. Wenn Sie dabei mit der Zeilennummer 71 für den vierten Posten beginnen und bis Titel Nummer 20 in Einerabständen fortfahren, gibt es auch mit der Zeilennummerierung keine Probleme. Ansonsten hilft natürlich ein einfaches »RENUM«.

Nachdem die Namen der einzelnen Posten eingelesen sind, wird die erste Liste aufgebaut. Das zugrundeliegende Schema kennen Sie ja schon von Bild 2. Die 20 Posten werden in einer Schleife ausgegeben. Zunächst wird die Schleifenvariable i in Zeile 200 mit einem ersten USING ausgegeben. Dies dient nur dazu, die einzelnen Zeilen exakt zu kennzeichnen, was Änderungseingaben erleichtert. Normalerweise würde der Schneider den Cursor nach diesem ersten PRINT-Kommando auf den Beginn der nächsten Zeile setzen und damit den nächsten Ausdruck eine Zeile tiefer starten. Durch das angehängte »;« bleibt er aber in der aktuellen Zeile. Als zweites PRINT USING treffen wir nun auf die formatierte Namensausgabe. Die Länge der Postenbeschreibung ist durch die Schrägstriche auf 13 Zeichen begrenzt. Wieder folgt ein »;«, das den Cursor in der aktuellen Bildschirmzeile festhält. In Zeile 220 wird dann der Wert dieses Postens ausgegeben. Da der Name nicht direkt an der Wertzuweisung kleben sollte, sind vor dem »DM« noch zwei Leerzeichen eingefügt. Die Werte der einzelnen Posten werden in der Variablen »sum« aufaddiert. Diese wird in der letzten Spalte unserer Ausgabe laufend in jeder Bildschirmzeile dargestellt. Dabei wurde allerdings noch das Steuerkommando »CHR$(24)« benutzt. Dieses bewirkt eine Reversdarstellung der letzten Spalte, die diese Werte noch einmal besonders hervorhebt. Damit nun nicht auch die zwei Trennleerstellen zwischen der vorletzten und der letzten Spalte revers dargestellt werden, haben wir diese vorsorglich schon mit der vorletzten PRINT USING-Anweisung (in Zeile 220) ausgegeben. PRINT USING übernimmt also keine Leerzeichen, die vor der Formatanweisung eingefügt sind. Auch danach stehende Leerstellen werden normal behandelt. Mit dem vierten USING-Kommando ist dann die Ausgabe der aktuellen Bildschirmzeile beendet.

Neben der Darstellung auf dem Bildschirm kann man aber auch einen Ausdruck des Haushaltsbuches erzeugen. Dazu dienen die Zeilen 300 bis 400. Verwendet man PRINT USING mit einem Drucker, sollten Sie sich allerdings über eine Tatsache nicht wundern. Wenn Sie, wie in unserem Programm, Daten mit mehreren PRINT USING-Anweisungen aneinanderketten, so wird die Zeile erst nach dem letzten PRINT, dann aber auf einen Schlag, ausgedruckt. Der Grund ist einfach. Intern formt der Computer die eingegebenen Werte natürlich in einen formatierten String um. Erst wenn dieser fertiggestellt ist, kommt die Druckerausgabe an die Reihe. So ist es nicht verwunderlich, daß der erste, bereits formatierte Teilausdruck, noch nicht auf dem Papier steht, wenn Sie zwischen zwei verbundenen PRINT USING-Kommandos noch eine zeitraubende Berechnung eingebaut haben.

Die beiden Listendarstellungsroutinen sind als Unterprogramme ausgeführt, die von der Hauptschleife in den Zeilen 440 bis 540 aufgerufen werden. Diese gibt vier Handlungsmöglichkeiten aus: den Druck der Liste, ihre Abspeicherung, das Laden der Listendaten und die Eingabe neuer Werte für einen Posten. Die Eingabe neuer Werte ist dabei denkbar einfach. Haben Sie eine neue Ausgabe verzeichnet, beispielsweise für 45.60 Mark getankt, so wählen Sie einfach Posten 3 und geben dann diesen Betrag ein. Der CPC addiert dann dazu den alten Wert und stellt die aktuellen Summen automatisch wieder dar.

Wie schon weiter oben gesagt, ist die hier beschriebene Routine aus Listing 2 lediglich eine Rumpfversion. Sie können natürlich die einzelnen Ausgabeposten zu Gruppen zusammenfassen und dann verschiedene Unterergebnisse bilden. Auch können Sie die Darstellung nach der Eingabe beschleunigen, indem Sie nicht jedesmal den gesamten Bildschirm neu aufbauen, sondern nur ab der geänderten Zeile alles neu ausgeben. (Carsten Straush/hg)

```
10 '********************                    [F28C]
20 '** Haushaltsbuch **                      [57C0]
30 '********************                    [2A90]
40 DIM n$(20):DIM n(20)                      [B030]
50 n$(1)="Lebensmittel"                      [1D04]
60 n$(2)="Koerperpflege"                     [F0CE]
70 n$(3)="Autokosten"                        [AD96]
80 n$(4)="IHRE WAHL..."                      [653C]
90 'n$=Ausgabenname n=Betrag                 [84CC]
100 'b=budget                                [6BE4]
110 GOSUB 130:GOTO 440                       [FF22]
120 '************************                [6440]
130 '** Darstellung Liste **                 [A3D2]
140 '************************                [A444]
150 CLS                                      [8230]
160 sum=0                                    [5DF2]
170 PRINT"Titel{14 SPACE}Betrag{6 SPACE}
    Summe"                                   [4BCE]
180 PRINT STRING$(39,"-")                    [C976]
190 FOR i= 1 TO 20                           [BAFA]
200 PRINT USING"##":i;                       [0C9C]
210 PRINT USING"\{11 SPACE}\";n$(i);         [E408]
220 PRINT USING"{2 SPACE}DM#####.##{2 SP
    ACE}";n(i);                              [33FA]
230 sum=sum+n(i)                             [DFDE]
240 PRINT CHR$(24);USING"DM#####.##":sum
    ;:PRINT CHR$(24)                         [547C]
250 NEXT                                     [74EC]
260 RETURN                                   [BB30]
270 '*****************                       [D100]
280 '**Druckausgabe**                        [B9B4]
290 '*****************                       [BB04]
300 sum=0                                    [ABEA]
310 PRINT#8,"Titel{14 SPACE}Betrag{6 SPA
    CE}Summe"                                [A7D4]
320 PRINT#8,STRING$(39,"-")                  [9E3C]
330 FOR i= 1 TO 20                           [4CF2]
340 PRINT#8,USING"##":i:                     [C174]
350 PRINT#8,USING"\{11 SPACE}\";n$(i);       [54E0]
360 PRINT#8,USING"{2 SPACE}DM#####.##{2
    SPACE}";n(i);                            [A2D2]
370 sum=sum+n(i)                             [84E8]
380 PRINT#8,USING"DM#####.##":sum            [32BA]
390 NEXT i                                   [3608]
400 RETURN                                   [CD28]
410 '**********************                  [599C]
420 '** Abfrageschleife **                   [1060]
430 '**********************                  [D9A0]
440 LOCATE 3,24:PRINT"Eingabe(1) SAVE(2)
     LOAD(3) Druck(4)"                       [B582]
450 z$=INKEY$:IF z$="" THEN 450              [C190]
460 IF ASC(z$)<49 OR ASC(z$)>53 THEN 450
                                             [CD5A]
470 IF z$="4" THEN GOSUB 300                 [B1BE]
480 IF z$="2" THEN CLS:OPENOUT"Haushalt"
    :FOR i=1 TO 20:PRINT#9,n(i):NEXT i:C
    LOSEOUT                                  [50DA]
490 IF z$="3" THEN CLS:OPENIN"Haushalt":
    FOR i=1 TO 20:INPUT#9,n(i):NEXT i:CL
    OSEIN                                    [3960]
500 IF z$<>"1" THEN 540                      [5AF2]
510 INPUT"Welche Zeile";z                    [CE60]
520 LOCATE 20,25:INPUT"Betrag +/-";b         [7356]
530 n(z)=n(z)+b                              [96E8]
540 GOSUB 130:GOTO 440                       [AD30]
```

Listing 2. Haushaltsbuch hilft Geld sparen

Fensterln mit System



Als eine der ersten Heimcomputer-Sprachen stellt das Locomotive-Basic von Schneider Windows zur Verfügung, die den Bildschirm in punkto Übersichtlichkeit revolutionieren. »Fensterln« Sie doch mit!

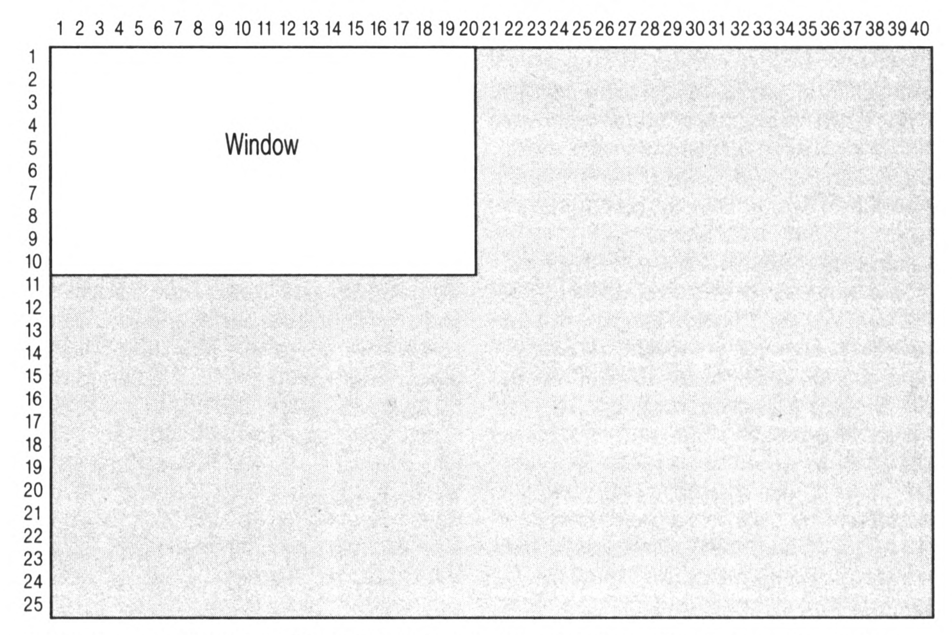


Bild 1. Das »WINDOW #1,1,20,1,10« im Bildschirm-Modus 1 (40 Zeichen pro Zeile)

In fast jeder Werbung von Schneider tauchen sie auf: die Windows. Worum handelt es sich dabei? Ein Window ist ein Bildschirmfenster. Der Computer legt dabei innerhalb des Gesamtbildschirms einen Bereich fest, den er wie einen eigenen kleineren Bildschirm behandelt. Wenn wir beispielsweise ein Textfenster in der linken oberen Ecke unseres Bildschirms definieren (siehe Bild 1) und festlegen, daß die Breite dieses Fensters 20 Zeichen betragen soll und seine Höhe zehn Zeilen, so springt der Cursor bei der Zeichenausgabe nach der Ausgabe des zwanzigsten Zeichens (von links ab gerechnet) in die nächste Zeile. Das Hochschieben der Zeilen (Scrollen) erfolgt in diesem Textfenster nun schon nach der zehnten Zeile. Im Endeffekt haben wir es bei einem Window also mit einem Bildschirm im Bildschirm zu tun.

Die Definition solcher Bildschirmfenster ist einfach: Man gibt einfach die Koordinaten für die Eckpunkte des Fensters an. Bezugsrahmen ist dabei immer der Gesamtbildschirm. Dieser besteht aus 25 Zeilen (Zeile 1 ist dabei die oberste) und je nach Modus 20, 40 oder auch 80 Zeichen je Zeile. In diesem Rahmen kann man dann bis zu 8 Windows definieren. Geben wir beispielsweise »WINDOW #1,1,20,1,10« ein, so wird ein Fenster mit den gerade beschriebenen Eigenschaften als Window Nummer 1 definiert. Die einzelnen Parameter des Kommandos sind dabei die Windownummer, gefolgt von der Angabe der linken und rechten Spalte des Windows, sowie der obersten und untersten Zeile.

Alle Befehle, die dieses Window adressieren, wirken nun nur noch auf den durch das Window abgedeckten Teil des Bildschirms. Dazu fügen wir nach einem Befehl, der das Fenster ansprechen soll, einfach die Window-Nummer mit dem Doppelkreuz (»#«) ein. Bei der Ausgabe von beispielsweise »PRINT #1,"Probetext"« erscheint das Wort »Probetext« in der aktuellen Zeile des Windows Nummer 1. Wenn in diesem Window noch keine Ausgaben erfolgt sind, ist dies wie beim Gesamtbildschirm die oberste Zeile des Windows. Geben Sie diesen Befehl nun zehnmal hintereinander aus, so scrollt das Window. Die oberste Zeile wird aus dem Textfenster herausgeschoben und verschwindet, unten wird eine neue angefügt.

Windows unterteilen den Bildschirm

Versuchen wir einmal einen Satz mit PRINT #1 auszugeben, der länger als 20 Zeichen ist (beispielsweise »PRINT #1,"Dieser Text ist laenger als 20 Zeichen"«). Der Satz wird hier nicht wie gewohnt in einer Zeile auf dem Schirm dargestellt, sondern nach 20 Zeichen in die nächste Windowzeile herübergezogen (siehe auch Bild 2).

Die Ausgabe von Zeichen und Zahlen an ein Textfenster erfolgt normalerweise mit PRINT #. Neben dem PRINT-Kommando dürfen wir aber auch noch eine Reihe anderer Text-Kommandos, wie das genaue Festlegen einer Position mit dem Befehl LOCATE oder das Löschen des (Teil-)Bildschirms mit CLS, benutzen. Allerdings beziehen sich alle Positionsangaben jetzt immer auf die linke obere Ecke des Windows. Daneben gehört zu jedem Kommando, das ein Textfenster ansprechen soll, natürlich immer die Window-Nummer.

»CLS #3« löscht das Window Nummer 3. »LOCATE #3,5,3,3« setzt den Cursor auf die zehnte Spalte der dritten Zeile dieses Windows. Angenommen, Fenster 3 wurde mit »WINDOW #3,10, 20,10,15« definiert, dann steht der Cursor jetzt auf den Koordinaten, die im Fenster 0 (Gesamtbildschirm) mit »LOCATE #0,14,12« angesprochen werden. Die Koordinaten von Window und Gesamtbildschirm unterscheiden sich also. Man kann demzufolge mit Hilfe der Windows den Gesamtschirm unterteilen. Wie dies geht, wollen wir uns nun anhand einiger Beispiele anschauen.

Als erstes versuchen wir, eine Tabellenausgabe mit Hilfe von Windows zu realisieren. Ein praktisches Beispiel dazu könnte eine tabellarische Wettkampfauswertung sein, beispielsweise die Punktetabelle der Fußball-Bundesliga. Eine solche Tabelle enthält bekanntlich drei Spalten. In der ersten findet man den Namen des Vereins, die zweite Spalte enthält das erreichte Punktergebnis, die dritte Spalte zusätzliche Angaben, in unserem Fall das Tor-Verhältnis.

Wir setzen nun voraus, daß uns die erforderlichen Angaben in Form von Datenfeldern zur Verfügung stehen. Die Variable n$ soll dabei den Namen enthalten. n$(1) steht dabei für den ersten Verein, n$(2) für den zweiten und so weiter. Wenn wir also auf unser Beispiel, die Fußball-Bundesliga, zurückkommen, wäre beispielsweise n$(1) = "1. FC Köln"; n$(2)="Bayern München" und so weiter. Nun brauchen wir noch ebensoviele Strings für das Punktverhältnis. Hier nehmen wir die Variable p$. p$(1) enthält also den Punktestand des ersten Vereins (rein fiktiv also beispielsweise einmal



»16:12«), p$(2) den des zweiten Vereins und so weiter. Das Torverhältnis soll nach demselben Prinzip in der Variablen t$ gespeichert sein.

Nachdem wir wissen, wo unsere Werte im Speicher stehen, können wir uns mit der Ausgabe beschäftigen. Die Namen sind in der ersten Spalte darzustellen. Dies ist kein besonderes Problem, da die erste Spalte ja am linken Bildschirmrand anfangen kann. Wir müßten also einfach beispielsweise in einer FOR-NEXT-Schleife (oder durch 20 PRINT-Befehle) die Namen untereinander ausgeben.

Schwieriger wird es bei der Ausgabe des Punktestandes. Hier muß die Darstellung ja in der Bildschirmmitte erfolgen. Auch dieses Problem kann man – mit ein wenig Gehirnakrobatik – noch im Rahmen des Gesamtbildschirms lösen. Wir müßten für jeden Verein festlegen, an welcher Stelle dessen Punktestand ausgegeben werden soll, dann den Cursor mit LOCATE auf diese Stelle setzen und dort den Punktestand ausdrucken. Dieses Verfahren ist jedoch wenig komfortabel. Einfacher geht es mit Windows.

Windows als Editierhilfe

Als erstes entscheiden wir, wie groß (breit) unsere Spalten sein sollen. Reservieren wir zunächst einmal 20 Zeichen Breite für den Namen in der ersten Spalte und je zehn Zeichen für die anderen beiden Angaben. Nun definieren wir drei Windows. Das erste soll für die Namensausgabe benutzt werden. Die Darstellung erfolgt dabei über die gesamte Höhe des Schirms. Wir legen deshalb also fest: »WINDOW # 1,1,20,1,25«.

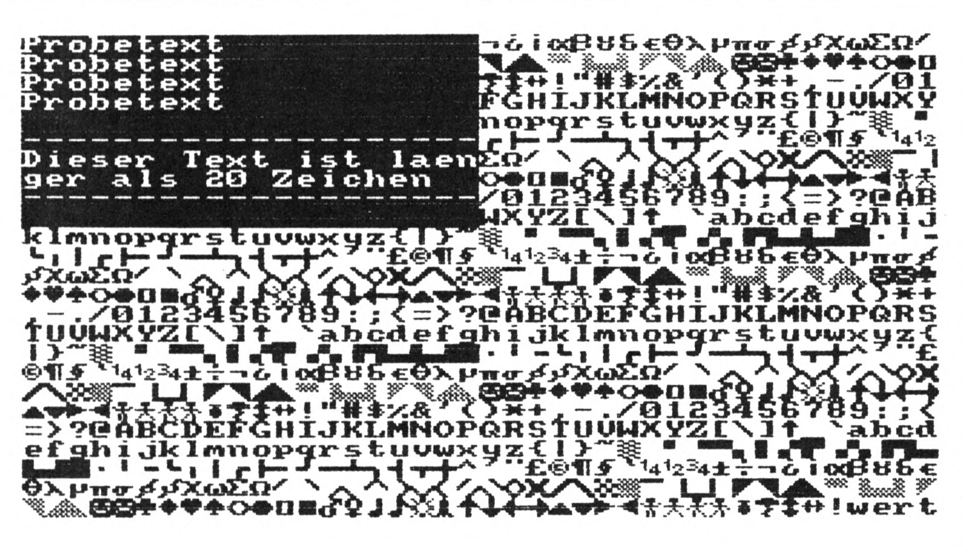


Bild 2. Windows machen den Bildaufbau übersichtlich

```
Verein 1        19: 2     7: 1     5 MODE 2
Verein 2        15: 6     1: 3     10 CLS
Verein 3         4:17     5:17     20 WINDOW#1,1,20,1,25
Verein 4        13: 8     1: 7     30 WINDOW#2,21,30,1,25
Verein 5         0:21     0:23     40 WINDOW#3,31,40,1,25
Verein 6        16: 5    24: 3     50 DIM n$(20),p$(20),z$(20)
Verein 7         9:12    11: 7     60 FOR i=1 TO 20
Verein 8        11:10     7:15     70 n$(i)="Verein"+STR$(i)
Verein 9         7:14     6: 3     80 NEXT i
Verein 10       13: 8    19:12     90 FOk i=1 TO 20
Verein 11       20: 1     4: 0     100 p=20*(RND(1))
Verein 12        2:19     1:12     110 p$(i)=DEC$((p,"##")+":"+DEC$((21-p,"
Verein 13       11:10    14: 7     ##")
Verein 14       15: 6    27: 4     120  z$(i)=DEC$((INT(RND(1)*2*p),"##")+"
Verein 15        9:12    11:18     :"+DEC$((INT(RND(1)*(20-p)*2),"##")
Verein 16        5:16     7: 1     130 NEXT i
Verein 17       20: 1    18: 0     140 FOR i=1 TO 20
Verein 18       11:10    11: 7     150 PRINT#1,n$(i)
Verein 19        2:19     4:25     160 PRINT#2,p$(i)
Verein 20       11:10     3:11     170 PRINT#3,z$(i)
                                   180 NEXT i
                                   190 WINDOW#0,41,80,1,25
                                   200 LIST
                                   Ready
                                   |copy
```

Bild 3. Listing und Programmablauf auf einen Blick

Das erste Textfenster reicht damit auf den Gesamtbildschirm bezogen von Spalte 1 bis Spalte 20. Analog dazu werden auch die anderen Fenster festgelegt: »WINDOW # 2,21,30,1,25« und »WINDOW # 3,31,40,1,25«

Die eigentliche Ausgabe ist nun problemlos. Das Programm aus Listing 1 geht näher darauf ein.

In den ersten Zeilen werden die drei verschiedenen Windows eröffnet und die Variablen mit fiktiven Werten geladen. Von Interesse ist erst der Bereich ab Zeile 140. In einer Schleife werden die einzelnen Werte auf die drei Textfenster verteilt. So einfach kann eine Tabelle mit Hilfe der Windows ausgegeben werden.

Ein besonderer Vorteil, den die Windowbenutzung vermittelt, liegt im eingebauten Schutz gegen Überschreiben durch spaltenfremde Informationen. Eine Ausgabe, die an ein Window gerichtet ist, kann nur diesen Bereich des Bildschirms verändern – nicht jedoch Informationen in der nächsten Spalte. Auch wenn Sie mit PRINT # 2 einen sehr langen »Zeichenbandwurm« ausgeben, kann dieser niemals auf die anderen beiden Fenster übergreifen.

Die gerade gezeigte Unterteilung des Bildschirms in mehrere Bereiche ist natürlich nicht auf die Listen- beziehungsweise Tabellenausgabe beschränkt. Sie läßt sich auch sehr gut bei Programmentwicklungen einsetzen. Ein interessantes Anwendungsfeld ist hierbei die Fehlersuche – das sogenannte Debugging.

Häufig ist es nützlich, wenn der Schirm neben der Bildschirmausgabe gleichzeitig auch die Zeilen, die diesen Ausdruck bewirken, zeigt. Wenn man nun im Modus 1, also mit 40 Zeichen pro Zeile, arbeitet, so ist dies natürlich nicht möglich. Es gibt jedoch einen kleinen Trick.

Auch der Gesamtbildschirm ist ein Window

Der CPC kann nicht nur Windows adressieren, er spricht sogar ausschließlich Windows an. Auch eine normale Ausgabe mit PRINT geschieht immer in einem Window, normalerweise dem Window # 0. Sie können dies relativ einfach überprüfen, indem Sie »WINDOW # 0,1,20,1,25« eingeben. Geben Sie nun mit PRINT einen Text aus, erfolgt nach 20 Zeichen der Sprung in die nächste Zeile. Wir haben also mit der Änderung von Window # 0 die Ausgabebreite für das »normale« PRINT verändert. Die üblichen Ausgaben mit PRINT greifen jetzt nur noch auf die linke Hälfte des Bildschirms zurück.

Dies allein gibt uns natürlich noch nicht die Möglichkeit, Ausgabe und Listing parallel darzustellen. Da wir nämlich nur noch 20 Zeichen Breite zur Verfügung haben, können wir unsere Ausgabepositionierung, die ja im Modus 1 auf 40 Zeichen Breite abgestellt ist, nicht überprüfen. Dieses Problem löst sich durch Umschalten in den nächsthöheren Modus, in unserem Fall Modus 2 (»MODE 2«), in dem 80 Zeichen pro Zeile zur Verfügung stehen. Wir können mit »WINDOW # 0,1,40,1,25« und »WINDOW # 1,41,80,1,25« den Schirm splitten.

Ist das kleine Beispielprogramm noch im Speicher, können Sie durch drei neue Programmzeilen das Listing ausgeben und parallel dazu mit RUN den Programmlauf kontrollieren. Die neuen Zeilen finden Sie in Listing 2, das Ergebnis in Bild 3.

Windows in Farbe

Bis jetzt haben wir immer mit farblich identischen Textfenstern gearbeitet. Gerade aber unterschiedliche Hinter- und Vordergrundfarben eröffnen in Zusammenhang mit den Windows interessante Perspektiven. Schauen wir uns zunächst an, wie die Farbe der Windows festgelegt ist. Das Prinzip ist dabei identisch mit der Farbgebung für den Gesamtbildschirm, der ja, wie wir gerade gesehen haben, auch nur ein Window darstellt. Als ersten Schritt belegen wir die Farbregister mit Hilfe von INK mit den benötigten Farbcodes. Dann bestimmen wir mit Hilfe von PEN und PAPER, welches Farbregister für die Vorder- beziehungsweise Hintergrundfarbe unseres Windows benutzt werden soll.

»PEN # 1,1« definiert zum Beispiel, daß für das Window mit der Nummer 1 die Farbe zum Schreiben benutzt werden soll, die in dem Farbregister 1 gespeichert wurde.

»PAPER # 2,3« gibt analog dazu an, daß die Hintergrundfarbe für Window Nummer 2 aus dem Farbregister Nummer 3 genommen werden soll. Mit CLS # 2 wird dann Window # 2 in die neue Hintergrundfarbe geändert.

Indem man eine neue Hintergrundfarbe definiert und mit dieser ein Window löscht, ist es möglich, mehrere Bildschirmbereiche und damit auch verschiedene Funktionen farblich zu trennen. Eine klare Unterscheidung von Ein- und Ausgaben verbessert beispielsweise die Arbeit mit einem Programm ungemein. In der Praxis hat sich dabei eine Dreiteilung bewährt, sozusagen ein Goldener Schnitt der Bildschirmaufteilung: Die obersten drei Zeilen sind dabei für den Titel reserviert.

```
10 CLS                                        [DDC6]
20 WINDOW#1,1,20,1,25                         [0212]
30 WINDOW#2,21,30,1,25                        [FB7C]
40 WINDOW#3,31,40,1,25                        [9684]
50 DIM n$(20),p$(20),z$(20)                   [F968]
60 FOR i=1 TO 20                              [2352]
70 n$(i)="Verein"+STR$(i)                     [FF7E]
80 NEXT i                                     [81A0]
90 FOR i=1 TO 20                              [3A58]
100 p=20*(RND(1))                             [9442]
110 p$(i)=DEC$((p,"##")+":"+DEC$((21-p,"
   ##")                                       [F07E]
120  z$(i)=DEC$((INT(RND(1)*2*p),"##")+"
   :"+DEC$((INT(RND(1)*(20-p)*2),"##")        [C214]
130 NEXT i                                    [23F8]
140 FOR i=1 TO 20                             [1DB0]
150 PRINT#1,n$(i)                             [6C1E]
160 PRINT#2,p$(i)                             [5926]
170 PRINT#3,z$(i)                             [AA3E]
180 NEXT i                                    [1202]
190 GOTO 190                                  [CF5A]
```

Listing 1. Tabellengestaltung mit Windows (hier wird Listing 2 eingesetzt)

```
5 MODE 2                                      [5C98]
190 WINDOW#0,41,80,1,25                       [57F4]
200 LIST                                      [5FDC]
```

Listing 2. Drei Befehle – und das Editieren wird ein Kinderspiel

```
10 INK 0,0:INK 1,11:INK 2,24:INK 3,6          [C2B6]
20 WINDOW#1,1,40,1,3                          [A5AE]
30 WINDOW#0,1,40,4,20                         [1912]
40 WINDOW#2,1,40,21,25                        [7380]
50 PAPER#0,0:PAPER#1,1:PAPER#2,1              [E1E6]
60 PEN#0,2:PEN#1,3:PEN#2,3                    [2B76]
70 CLS:CLS#1:CLS#2                            [EF94]
80 PRINT#1,CHR$(10)+"{7 SPACE}Hier kann
   der Titel stehen                           [87D2]
90 FOR i=1 TO 1000:NEXT                       [FB08]
100 PRINT"{2 SPACE}Eine Ausgabe auf dem
   Hauptbildschirm"                           [F620]
110 FOR i=1 TO 1000:NEXT                      [FB5A]
120 PRINT#2,"So koennte eine Eingabeabfr
   age aussehen"                              [9756]
130 GOTO 130                                  [C542]
```

Listing 3. Der dreigeteilte Arbeitsbildschirm

```
10 MODE 0                                     [DBEC]
20 CLS                                        [DEC8]
30 INK 1,4:INK 2,12:INK 3,2:INK 4,9           [5B6A]
40 WINDOW#1,1,3,21-2*4,21                     [098E]
50 PAPER#1,1                                  [2A5C]
60 CLS#1                                      [9178]
70 WINDOW#1,5,7,21-4*4,21                     [81A8]
80 PAPER#1,2                                  [2B64]
90 CLS#1                                      [947E]
100 WINDOW#1,9,11,21-5*4,21                   [AE5C]
110 PAPER#1,3                                 [2ABA]
120 CLS#1                                     [55D2]
130 WINDOW#1,13,15,21-4*4,21                  [75BE]
140 PAPER#1,4                                 [B3C2]
150 CLS#1                                     [58D8]
160 GOTO 160                                  [EE4E]
```

Listing 4. Balkendiagramme sind mit Windows leicht zu gestalten

```
10 REM *****************                      [ACF2]
20 REM ** Farbfelddemo **                     [AA52]
30 REM *****************                      [48F6]
40 INK 0,0:INK 1,24:INK 2,5:INK 3,6:INK
   4,18                                       [CD6C]
50 INK 5,12:INK 6,7:INK 7,10:INK 8,2:INK
   9,24                                       [6BF2]
60 PAPER 0:MODE 0:BORDER 0                    [9B8A]
70 PEN 2:LOCATE 3,5:PRINT"Tastaturbelegu
   ng"                                        [D41C]
80 FOR i= 0 TO 9 STEP 3                       [8FA0]
90 FOR k= 0 TO 3 STEP 3                       [D19A]
100 WINDOW#1,5+i,7+i,10+k,12+k                [2322]
110 PAPER #1,(2+i/3+4*k/3):CLS#1              [9012]
120 NEXT k,i                                  [7424]
130 LOCATE 6,11:PRINT"Q"                      [3D58]
140 LOCATE 9,11:PRINT"W"                      [236C]
150 LOCATE 12,11:PRINT"E"                     [0E9E]
160 LOCATE 15,11:PRINT"R"                     [42C0]
161 LOCATE 6,14:PRINT"A                       [5102]
162 LOCATE 9,14:PRINT"S                       [922E]
163 LOCATE 12,14:PRINT"D                      [E466]
164 LOCATE 15,14:PRINT"F                      [FA72]
170 GOTO 170                                  [F152]
```

Listing 5. Der Bildschirm gibt Informationen über die Tastatur

Dieser Bereich ist besonders bei menügesteuerten Programmen, also Programmen mit meist vielen verschiedenen Funktionen, die durch einen Tastendruck angesprochen werden, sehr nützlich. Durch die Anzeige der gerade ausgewählten Funktion weiß man immer, in welchem Programmteil man sich befindet.

Nach demselben Prinzip reservieren wir im unteren Teil des Schirms - je nach Bedarf - drei bis fünf Zeilen für die Kommunikation mit dem Computer, also die Eingabe von Texten, Daten und dem Ausdruck von Fragetexten. Der Rest des Bildschirms bleibt als Arbeitsfeld des CPC, wo Daten oder Grafiken ausgegeben werden können.

Mit Hilfe der Windows können wir eine solche Unterteilung relativ einfach realisieren. Listing 3 verdeutlicht das Prinzip.

Der Titelbereich wurde mit Window #1 bezeichnet. Ein PRINT #1 - beziehungsweise CLS #1 - gibt also den Titel aus und löscht das Titelwindow. Analog dazu ist auch der Eingabebereich definiert. Titel und Eingabe sind dabei mit derselben Farbkombination belegt (rote Schrift auf blauem Grund - für die Besitzer eines grünen Monitors dunkelgrün auf mittelgrün). Dazu werden zunächst die Farbregister 0 bis 3 auf die Farben Schwarz (Farbcode 0), Mittelblau (Farbcode 11), Gelb (Farbcode 24) und Rot (Farbcode 6) gesetzt, bevor diese Registerdefinitionen auch für die Definition der Vorder- und Hintergrundfarben Verwendung finden (Zeilen 50, 60). Nach den CLS-Befehlen in Zeile 70 sind die drei Textfenster auch durch farbliche Trennung direkt zu erkennen. Der Hauptbildschirm (Window #0) wurde um den Titel- beziehungsweise Eingabebereich verringert. Er umfaßt jetzt nur noch die Zeilen 4 bis 20. Wundern Sie sich also nicht, wenn Sie beispielsweise mit LIST das Programm ausdrucken und dann feststellen, daß im oberen und unteren Teil des Bildschirms ein paar Zeilen nicht mehr benutzt werden. Dies ist ein Nebeneffekt unserer Operation. Denn bei jeder normalen Ausgabe, so auch bei LIST, spricht der Schneider Window #0 an, und genau dieses haben wir gerade verkleinert.

Farblich unterlegte Fenster kann man aber nicht nur für die Unterteilung des Bildschirms im Rahmen eines menügesteuerten Programmes benutzen. Sie sind auch ein nützliches Hilfsmittel im grafischen Bereich. Zwei kleine Programme zeigen dies.

Mit Hilfe der Textfenster kann man sehr einfach eine Balkengrafik (Bild 4) zaubern. Betrachten wir uns zunächst das Prinzip, nach dem dies funktioniert. Sie wissen bereits, daß sich Windows nicht nur zur vertikalen Einteilung des Bildschirms eignen, sondern auch, wie bei unserer Editierhilfe, zur Unterteilung in der Horizontalen. Das geht natürlich auch mit mehreren Windows nebeneinander.

Während wir bei der Editierhilfe immer den gesamten Bildschirm als Windowhöhe gewählt hatten, sieht das bei der Grafik etwas anders aus. Je nachdem, welchen Wert wir darstellen wollen, setzen wir die Obergrenze unterschiedlich. In unserem Beispiel stellen wir vier Werte dar, die 2,4,5 und wieder 4 betragen. Wir können hier problemlos mit Windows arbeiten. Zunächst wird die Null-Linie festgelegt, der Einfachheit halber wählen wir Zeile 21. Damit stehen die darüberliegenden 20 Zeilen zur Darstellung zur Verfügung. Als nächstes definieren wir die Einheiten. Dies ist einfach, da der größte darzustellende Wert 5 ist. Mit maximal 20 Zeilen können wir je Einheit also vier Zeilen reservieren.

Bei der Window-Definition haben wir allerdings ein kleines Problem. Die Zeilennummern werden nach unten zu höher (der Punkt (1,1) liegt ja oben links), während unsere mit den Windows gezeichneten Balken mit zunehmenden Werten nach oben, also in der umgekehrten Richtung, wachsen sollen. Durch eine geeignete Window-Definition läßt sich dieses Problem jedoch leicht lösen. Für den ersten Wert (2) definieren wir: »WINDOW #1,1,3, 21-2 * 4,21«, »PAPER #1,1« und »CLS #1«.

Die Zahl 21 ist dabei unsere Nulllinie, 21-2 Einheiten * 4 Zeilen je Einheit stellt die richtige Größe für den Wert 2 dar. Die Breite unseres Windows ist dabei beliebig. Hier wurden drei Spalten verwendet (Spalte 1 bis Spalte 3; Parameter 2 und 3 der Window-Definition). Man kann aber natürlich auch auf fünf Spalten je Wert wechseln. Die maximale Breite unserer Balken hängt nur von der Anzahl der darzustellenden Werte ab.

Bis jetzt haben wir einen einzelnen Wert dargestellt. Die Ausgabe der anderen Zahlen läuft aber dazu analog ab. Wir müssen nur die Spaltenparameter ändern und den neuen Wert einsetzen. Zum Ausdruck können wir dabei dasselbe Window noch einmal verwenden. Die alte Grafik bleibt auch ohne dieses Fenster auf dem Schirm erhalten. Beläßt man die Breite bei drei Spalten, sehen die zugehörigen Befehle so aus: »WINDOW #1,4,6,21-4 * 4,21«, »PAPER #1,1« und »CLS #1«.

Falls die Hintergrundfarbe schon feststeht, kann das PAPER-Kommando natürlich entfallen. Nach demselben Prinzip können Sie nun auch die anderen beiden Balken ziehen. Listing 4 gibt das vollständige Programm wieder. Variieren Sie ruhig einmal die einzelnen Parameter und schauen Sie sich an, was passiert. Als kleines Übungsbeispiel können Sie ein Programm schreiben, das unterschiedliche Werte als Balkendiagramm darstellt und dabei selbst den Maßstab (Zeilen pro Einheit) festlegt. Wenn Sie dabei größere Probleme haben, finden Sie eine Lösung für solch ein Programm im Schneider-Sonderheft Ausgabe 2/85.

Zum Abschluß - und als weitere Anregung zum Experimentieren mit den Windows - kommt noch ein ganz anderer Anwendungsbereich zur Sprache: die grafische Gestaltung des Bildschirms. Wenn man ein Programm schreibt, das viele verschiedene Funktionen auf Tastatur legt (beispielsweise ein Musikprogramm, das eine Orgel simuliert) so ist es erstrebenswert, die Tastaturbelegung permanent parat zu halten. Nun kann man natürlich den Bildschirm mit endlosen Erklärungstexten übersäen nach dem Prinzip: Die »Q«-Taste leistet dies und das etc. Besonders anwenderfreundlich ist diese Methode aber nicht. Besser bildet man die Tastatur auf dem Schirm als Grafik ab. Wie dies geht, zeigt Listing 5. Das Prinzip ist ähnlich der grafischen Darstellung, nur daß wir hier mit konstanten, verschiedenfarbigen Feldern arbeiten. Tippen Sie das Programm ruhig einmal ein, fügen Sie zwischen den einzelnen Zeilen ein STOP ein und lassen Sie sich überraschen, was man mit den Windows noch alles machen kann. Viel Spaß dabei!

(Carsten Straush/hg)

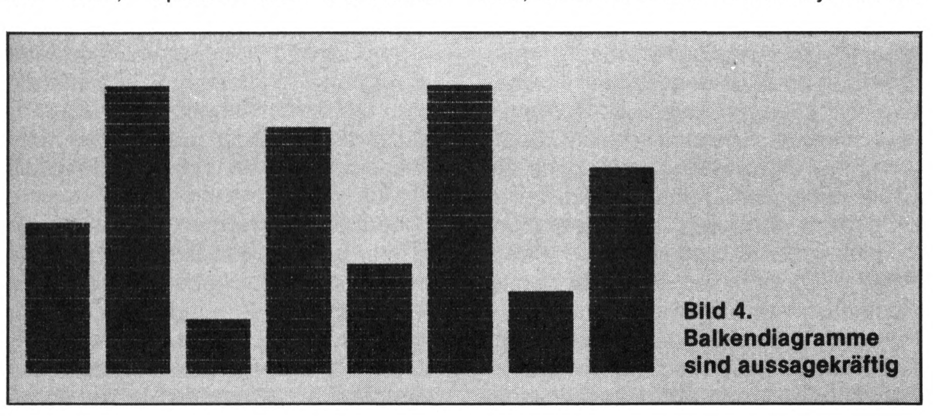


Bild 4. Balkendiagramme sind aussagekräftig

Dem Datenrecorder aufs Bit geschaut



Der Kassettenrecorder des CPC ist ein preiswertes Speichermedium. Was die Übersichtlichkeit über gespeicherte Dateien angeht, ist ihm jedoch jedes Diskettenlaufwerk weit voraus. Wir helfen ihm auf die Sprünge!

Der Datenrecorder des CPC 464 ist ein billiges, aber trotzdem – zumindest für den Anfänger – völlig ausreichendes Speichermedium. Neben Lesefehlern und der, speziell bei längeren Programmen, etwas langweiligen Speicherung, ergeben sich zwei Hauptprobleme. Ein einfacher Kassettenrecorder, wie der des Schneider, kennt kein Directory. Man kann daher Programme weder gezielt ansteuern noch gegen Überschreiben schützen. Daneben darf man Programmnamen mehrfach verwenden, was die Übersicht nicht gerade fördert. Als Ergebnis, vor allem wenn man mit mehreren Kassetten jongliert, dauert die Suche teilweise länger als der eigentliche Ladevorgang. Manchmal stellt man auch erst nach dem Schreiben des ersten Blocks fest, daß an dieser Stelle noch etwas gespeichert war, was man eigentlich aufheben wollte. Und dann ist es überschrieben und verloren.

Nun gibt es natürlich eine Reihe einfacher Hilfen. Man sorgt erstens dafür, daß jeder Programmname nur einmal verwendet wird und schreibt sich diesen mit Zählerstand und so weiter sorfältig auf. Ferner spult man jedesmal an den Bandanfang zurück und sucht dann mit Hilfe des Zählwerks exakt den Programmstart heraus. Mit guten Aufzeichnungen ist dies kein Problem. Hat man aber einmal den Inhaltszettel verloren, so beginnt die Sucherei wieder von vorn.

Mit CAT kann man von jeder Kassette einen Auszug auf den Schirm bringen und diesen dann abschreiben oder ausdrucken.

Dies ist jedoch ziemlich nervtötend und langwierig. Und da wir nicht auch noch der Buchhalter unseres Computers sein möchten, könnte die Maschine solche Aufzeichnungen ja selbst machen.

Es gibt eine ganze Reihe weiterer Informationen, die auf Band abgespeichert werden. Der Computer liest sie zwar auch ein, serviert sie uns aber bei einer einfachen CAT-Analyse nicht. Neben der Anzahl der abgespeicherten Bytes ist es besonders bei Maschinencode-Programmen interessant zu wissen, ab welcher Position diese im Speicher liegen. Bei selbststartenden Routinen benötigen wir daneben natürlich auch noch die Aufrufadresse. Ohne diese Angaben kann eine Maschinencode-Routine nicht lauffähig kopiert werden. Diese Informationen soll uns der Schneider auch mit angeben. Für die Ausgabe wählen wir zwei verschiedene Varianten. Die eine soll uns einen großen Ausdruck mit allen verfügbaren Informationen auf dem Drucker geben. Daneben wäre es günstig, wenn auch eine Kurzfassung (Dateiname, -typ und Blockzahl) auf Etiketten ausgedruckt werden könnte. Klebt man diese Etiketten dann auf die entsprechenden Kassetten, so hat man jederzeit eine optimale Übersicht.

Informationen im Programmkopf

Neben der praktischen Ausführung des Programms müssen wir uns noch damit beschäftigen, woher wir die für den Ausdruck benötigten Informationen erhalten. Alle wichtigen Werte sind im ersten Teil jeder Bandaufzeichnung, dem sogenannten Kopf oder Header, enthalten. Dieser Inhalt wird vor der Aufzeichnung jedes Blocks abgespeichert. Der Schneider ordnet anhand dieser Informationen die einzelnen Blöcke dem gesamten Programm zu. Nur so ist es beispielsweise möglich, daß bei einem Lesefehler in einem Block nur dieser wiederholt werden muß und das gesuchte Programm trotzdem richtig eingelesen werden kann. Beim Lesen wird der Header in einen eigens für diesen Zweck im Computer reservierten Speicherbereich abgelegt. Er ist 64 Byte lang und liegt für das Lesen einer Datei, unabhängig davon, ob es sich um Daten oder Programme handelt, ab der Adresse B88C hex vor. Parallel dazu existiert noch ein zweiter Speicherbereich von gleicher Länge, in dem der Schneider die Werte ablegt, die Sie ihm mit dem Ladebefehl übergeben haben. Wenn Sie also beispielsweise ein Programm mit Namen »Analyse« laden wollen und dazu LOAD »ANALYSE« eingeben, so wird dieser Name zunächst ab der Adresse 47111 (oder B807 hex) abgelegt. Findet der CPC beim Lesen ein Programm auf dem Band, so liest er zunächst dessen Header ab Adresse 47244 (B88C hex) ein. Dann wird verglichen, ob der eingelesene Kopf mit dem gewünschten Programm übereinstimmt. Erst danach werden die nachfolgenden Daten gelesen. Sie können dieses Vorgehen Schritt für Schritt nachvollziehen, indem Sie das nachfolgende kleine Experiment durchführen. Nehmen Sie ein beliebiges Programm XYZ und geben Sie »LOAD"XYZ"« ein. Statt auf PLAY zu drücken, brechen Sie jedoch den Programmlauf ab und schauen sich mit »FOR i=47111 TO 48000:?i, PEEK(i), CHR$(PEEK(i)):NEXT i« den Speicher an. In den ersten Speicherstellen finden Sie den Namen und eine 1 als Kennzeichen dafür, daß der erste Block eingelesen werden soll. Der restliche Speicher ist, falls Sie noch keine Schreib- oder Leseoperation (SAVE, OPENOUT beziehungsweise LOAD, OPENIN) ausgeführt haben, weitgehend leer. Wiederholen Sie nun das LOAD-Kommando. Brechen Sie aber erst nach dem Einlesen des ersten Blocks die Anweisung ab. Wenn Sie die FOR-TO-Schleife noch einmal eingeben, so finden Sie den Namen Ihres Programmes wie beschrieben ab der Adresse 47244. Dieser Programmname und die nachfolgenden Bytes, auf die wir gleich noch zu sprechen kommen, stellen eine Kopie des Kopfsatzes der Datei dar, die gerade vom Band gelesen wurde. In unserem Fall sind die beiden Namen identisch. Das Programm wird also geladen. Hätten wir aber beispielsweise als Programmnamen ANNA eingegeben und auf den Anfang von XYZ gespult, so würden wir »ANNA« ab 47111 sehen; »XYZ« stände ab 47244. In diesem Fall bestünde keine Übereinstimmung und der CPC würde das Band so lange durchsuchen, bis er einen Namen findet, bei dem sich eine Übereinstimmung ergibt.

Neben dem Namen, für den die ersten 15 Byte reserviert sind, enthält der Band-Header jedoch noch eine Menge anderer Informationen, die für uns sehr nützlich sind. Das Byte 18 beinhaltet eine Codierung für den Dateityp. Bei CAT finden Sie den Datei-

typ durch verschiedene Sonderzeichen repräsentiert. Das »$« ist der Code für ein Basic-Programm. »%« steht für ein geschütztes Basic-Programm. Bei »&« handelt es sich um eine Binär-Datei, normalerweise also um ein Maschinencode-Programm. Da sich auch Maschinencode-Routinen schützen lassen, existiert ein weiterer Code »'« als Symbol für eine geschützte Binär-Datei. Ein Sternchen (»*«) symbolisiert schließlich eine ASCII-Datei. Diese Speicherart verwendet der Schneider, um Daten abzulegen, beispielsweise bei OPENOUT. Für uns sind nun noch vier weitere Speicherstellen von Interesse. Die Bytes 24 und 25 des Headers enthalten die totale Länge der Datei in Bytes und bei Maschinencode-Programmen steht die Einsprungadresse, ab der der CPC das Programm automatisch startet, in den Bytes 26 und 27.

Damit haben wir nun alle notwendigen Informationen parat, um unser Kassettenanalyse-Programm zu schreiben. Die eigentliche Analyse stellt keine besonderen Probleme dar.

Doppelt soviel Information auf halbem Platz

Mit »OPENIN "!"« wird die jeweils nächste Datei auf dem Band gelesen (Zeile 480). In den folgenden Zeilen werden dann aus den einzelnen Header-Bytes die benötigten Informationen ausgelesen. Die einzige nicht direkt abgespeicherte Angabe, nämlich die Blockzahl, berechnet sich, indem man berücksichtigt, daß jeder Block maximal 2048 Byte aufnehmen darf. Aus der totalen Länge (Bytes 24 + 25) läßt sich daher durch Umformung die Blockzahl leicht nachrechnen. Wie dies geschieht, sehen Sie in den Zeilen 560 und 570.

Nachdem alle Ausgaben bereitstehen, werden sie zu dem String pr$ zusammengefügt. Dieser enthält je nach Ausgabegerät (Bildschirm oder Drucker) und Ausgabezweck (Etiketten- beziehungsweise Volldruck) unterschiedliche Angaben. Dabei ist zu sagen, daß bei der Druckoption auch noch jeweils eine parallele Ausgabe auf dem Schirm stattfindet. Die Zeichen werden dabei in gewohnter Größe dargestellt. Eine Ausnahme dieser Regel bildet der Etikettendruck. Da auf einem Etikett nur wenig Platz ist, sollen hier die Buchstaben möglichst klein erscheinen. Hat man beispielsweise sieben Titel auf einem Band gespeichert, so ist ein 3,5 cm hohes Etikett bei normaler Schriftgröße bereits voll. Dies ist, speziell wenn man viele kurze Programme hat, nicht gerade üppig.

Das Programm arbeitet deshalb mit einem Trick. Fast jeder gebräuchliche Nadeldrucker verfügt über zwei Druckoptionen: Superscript und

```
10 '******************************        [E828]
20 '** Cassettenanalyse CPC 464 **        [B8B8]
30 '**     by Carsten Strausch    **      [F84E]
40 '******************************        [672E]
50 '********************************      [2484]
60 '** Definition Druckerstrings **       [A730]
70 '********************************      [A488]
80 ps$(0)=CHR$(27)+"S1"                   [E0F6]
90 ps$(1)=CHR$(27)+"S0"                   [BEF8]
100 ps$(2)=CHR$(27)+"T"                   [A8EC]
110 ON BREAK GOSUB 890                    [346A]
120 MODE 1:BORDER 0                       [04E2]
130 INK 0,0:INK 1,24:INK 2,11:INK 3,21    [5076]
140 WINDOW#1,1,40,1,5:WINDOW#0,1,40,6,25
                                          [83A6]
150 WINDOW#3,1,40,20,25                   [00E4]
160 PAPER 0:PEN 1:PAPER#1,0:PEN#1,2       [F03C]
170 LOCATE#1,10,2: PRINT#1," Cassettenan
    alyse 464 "                           [CFDE]
180 LOCATE 4,15:PRINT"Soll die Ausgabe p
    aralell auf dem"                      [208A]
190 LOCATE 10,16:PRINT"Drucker erfolgen
    j/n?"                                 [7F72]
200 z$=INKEY$:IF z$="" THEN 200           [5774]
210 IF LOWER$(z$)="j" THEN flag=1         [F2C0]
220 IF LOWER$(z$)="n" THEN flag=0         [5DC8]
230 IF LOWER$(z$)<>"j" AND LOWER$(z$)<>"
    n" THEN 200                           [8308]
240 CLS                                   [8130]
250 PRINT#3,"Bitte spulen Sie das Band a
    uf den Anfang und druecken Sie eine
    beliebige Taste!"                     [C5B2]
260 CALL &BB18                            [600E]
270 IF flag=0 THEN 450                    [86EE]
280 CLS#3:PRINT#3                         [2D1E]
290 PRINT#3,"Bitte schalten Sie den Druc
    ker ein. DannTastendruck!"            [ADA8]
300 CALL &BB18                            [9504]
310 CLS#3:PRINT#3                         [4E12]
320 PRINT#3," Wie soll die Druckerausgab
    e erfolgen?"                          [2AE4]
330 PRINT#3                               [9932]
340 PRINT#3,"{8 SPACE}als Etikettendruck
     (1)"                                 [1AA4]
350 PRINT#3,"{8 SPACE}auf Endlospapier{3
     SPACE}(2)"                           [F960]
360 z$=INKEY$:IF z$="1" THEN eti=1 ELSE
    IF z$="2" THEN eti=0 ELSE 360         [B032]
370 IF eti=0 THEN PRINT#8,"Dateiname{7 S
    PACE}Typ{2 SPACE}Bloecke{2 SPACE}Byt
    es{2 SPACE}Anfang{2 SPACE}Ansprung"   [53B2]
380 IF eti=0 THEN PRINT#8,STRING$(65,"-"
    )                                     [1BE0]
390 PRINT#1                               [D73A]
400 PRINT#1,"Name{10 SPACE}Typ Bl. Bytes
     Anfang Start"                        [BE7E]
410 '********************                 [BA48]
420 '** Analyseroutine **                 [D430]
430 '********************                 [304C]
440 CLS                                   [9134]
450 ON ERROR GOTO 880                     [B032]
460 |TAPE                                 [8FC0]
470 n$=""                                 [EF9C]
480 OPENIN"!"                             [D7D4]
490 adr=&B88C                             [4198]
500 FOR i=adr TO adr+15                   [7588]
510 m=PEEK(i):IF m=0 THEN m=32            [0300]
520 n$=n$+CHR$(m)                         [D504]
530 NEXT i                                [4800]
540 typ$=CHR$(PEEK(adr+18) MOD 16 +36)    [768A]
550 laenge=PEEK(adr+24)+256*PEEK(adr+25)
                                          [2DA4]
560 bloecke=INT(laenge/2048)              [05E4]
570 IF bloecke*2048<laenge THEN bloecke=
    bloecke+1                             [F524]
580 ansprung=PEEK(adr+26)+256*PEEK(adr+2
    7)                                    [CAB6]
590 anfang=PEEK(adr+21)+256*PEEK(adr+22)
                                          [0B9E]
600 IF flag=0 THEN 790                    [E6F6]
610 IF eti=1 THEN 700                     [6238]
620 pr$=n$+" "+typ$+"{5 SPACE}"+RIGHT$("
    {2 SPACE}"+MID$(STR$(bloecke),2),2)   [87E4]
630 pr$=pr$+"{4 SPACE}"+DEC$((laenge,"##
    ####")+"{2 SPACE}&"+HEX$(anfang,4)    [4CB0]
640 IF ansprung=0 THEN pr$=pr$+"{3 SPACE
    }keine" ELSE pr$=pr$+"{3 SPACE}&"+HE
    X$(ansprung,4)                        [FEA2]
650 PRINT#8,pr$                           [97AA]
660 GOTO 790                              [126A]
670 '********************                 [4458]
680 '** Etikettendruck **                 [3826]
690 '********************                 [9E5C]
700 pr$=n$+" "+typ$+" "+RIGHT$("{2 SPACE
    }"+MID$(STR$(bloecke),2),2)           [69E2]
710 IF up=1 THEN up=0 ELSE up=1           [286E]
720 PRINT#8,ps$(up)+pr$;                  [2BEC]
730 IF up=1 THEN FOR i=1 TO 21:PRINT#8,C
    HR$(8);:NEXT                          [23B6]
740 IF up=0 THEN PRINT#8                  [9726]
750 GOTO 790                              [D16A]
760 '************************             [C254]
770 '** Bildschirmausgabe **              [774C]
780 '************************             [8258]
790 pr$=n$+typ$+" "+RIGHT$("{2 SPACE}"+M
    ID$(STR$(bloecke),2),2)               [5FD6]
800 pr$=pr$+" "+DEC$((laenge,"######")+"
    {2 SPACE}&"+HEX$(anfang,4)            [CDEE]
810 IF ansprung=0 THEN pr$=pr$+" keine"
    ELSE pr$=pr$+" &"+HEX$(ansprung,4)    [43A0]
820 PRINT pr$;                            [C850]
830 CLOSEIN                               [5090]
840 GOTO 470                              [E060]
850 '********************                 [4658]
860 '** ERROR Handling **                 [F638]
870 '********************                 [5C5C]
880 RESUME NEXT                           [06E0]
890 PRINT#8,ps$(2):END                    [01E0]
```

Listing. Analysieren Sie Ihre Kassettendateien

Subscript. Es handelt sich dabei um verkleinerte Zeichen, die nur die halbe Höhe eines normalen Symbols umfassen. Ein Superscript-Zeichen wird dabei in der oberen Hälfte einer Zeile ausgedruckt. Bei Subscript erfolgt die Ausgabe in der unteren Hälfte. Normalerweise werden diese verkleinerten Zeichen zur Angabe von Indices verwendet. Man kann sie jedoch auch für

So werden die Druckercodes angepaßt

die normale Schriftausgabe benutzen. In unserem Fall können wir dann durch Übereinanderdrucken von zwei Ausgaben in derselben Zeile – die erste mit Superscript, die zweite mit Subscript – die doppelte Anzahl (bis zu 15 Titel) auf einem Etikett unterbringen. Normalerweise reicht das aus.

Damit Sie gegebenenfalls das Programm an Ihren Drucker anpassen können, noch ein paar Worte zur Ausführung.

Zum Umschalten auf Subscript beziehungsweise Superscript benötigt der Drucker bestimmte Steuerbefehle. Eine solche Befehlssequenz beginnt normalerweise mit dem Code CHR$(27). Er sagt dem Drucker, daß die folgenden Angaben als Steuerbefehle zu betrachten sind, also nicht gedruckt werden sollen. Danach stehen noch ein oder zwei weitere Zeichen, die die Umschaltung auf die Schriftarten bewirken. Im Programm werden diese Kontrollsequenzen am Anfang in die Strings ps$(0), ps$(1) und ps$(2) geladen. ps$(0) schaltet dabei Subscript ein, ps$(1) Superscript. ps$(2) schaltet beide Schrifttypen aus und damit wieder auf normale Schriftgröße zurück. Die hier angegebenen Sequenzen entsprechen der Centronics-Norm, die die meisten Drucker verstehen. Falls Ihr Drucker andere Codes erwartet, müssen Sie nur diese Strings ändern. Einem weiteren Problempunkt bildet noch möglicherweise die Rücksteuerung des Druckkopfes. Wenn die obere Hälfte einer Zeile geschrieben wurde, muß danach von Superscript auf Subscript umgeschaltet (was ganz einfach mit Hilfe der Steuercodes geschieht) und gleichzeitig der Druckkopf wieder auf den Anfang der Zeile zurückgesetzt werden. Dazu gibt es mehrere Verfahren. Hier wurde eine allgemein bewährte Variante gewählt. Mit dem Drucker-Kommando <BACKSPACE> (CHR$(8)) wird der Druckkopf um eine Stelle zurückbewegt. Führt man diesen Befehl 21mal hintereinander aus – die auszugebenden Strings sind genau 21 Zeichen lang – so kommt man wieder zum Anfang der Zeile zurück. Sie finden das Kommando zum Rücksetzen in Zeile 730. Vielleicht verfügt Ihr Drucker auch über einen Code, der es erlaubt, den Druckkopf direkt auf den Zeilenanfang zurückzusetzen. Dann können Sie sich die FOR-TO-Schleife sparen und direkt diesen Code an den Drucker ausgeben. Nähere Informationen dazu finden Sie in Ihrem Druckerhandbuch.

Universeller Etikettendruck

Zum Abschluß nun noch ein Tip: Die hier benutzte Art des Etikettendrucks eignet sich natürlich nicht nur für die Katalogisierung von Datenkassetten. Man kann damit nach derselben Methode mit einigen kleinen Änderungen auch Ordnung in seine Musikkassetten- und Videobandsammlung bringen. (Carsten Straush/hg)

Links herum, rechts herum



Vielleicht wollten Sie immer schon eines der beliebten Labyrinth-Spiele programmieren? Oder aber, Sie interessiert die logische Grundlage? Hier erfahren Sie Näheres dazu.

Als Ausgangspunkt für eigene Spiele dienen diese Basic-Routinen: Sie erzeugen beliebig große Labyrinthe nach Maß, bei denen wirklich nur ein einziger Weg hinausführt.

Wenn Sie einen Computer ein Labyrinth erzeugen lassen wollen, stehen Sie vor der schwierigen Aufgabe, dem Computer den Labyrinth-Aufbau genauestens zu erklären. Der Zufall muß dabei gerade soweit eingeschränkt werden, daß zwar bei jeder Berechnung ein anderes Labyrinth entsteht, es aber andererseits wirklich nur einen Weg durch das Labyrinth gibt (und nicht etwa mehrere oder gar keinen).

Eine Lösung ist so möglich:

Schritt 1:

Zunächst soll das Labyrinth nur aus einem einzigen Feld bestehen, das irgendwo nahe der Mitte des endgültigen Labyrinthes liegt. Dieses Feld hat vier bisher unbelegte Nachbarfelder (Bild 1). Unbelegte, aber einem belegten Feld benachbarte Felder, werden ab jetzt »Randfelder« genannt.

Schritt 2:

Jetzt wird ein beliebiges der bisherigen Randfelder besetzt und mit einem der schon belegten Nachbarfelder verbunden. Dadurch entstehen wieder ein paar neue Randfelder (Bild 2).

Schritt 3:

Nun wird wieder ein beliebiges der sechs (alten und neuen) Randfelder besetzt und mit einem der bestehenden Felder verbunden. Die Nachbarfelder des neuen Feldes werden ebenfalls zu Randfeldern erklärt (Bild 3). Waren beim ersten Schritt nur vier Verbindungen möglich, so sind es jetzt schon sechs.

Schritt 4:

Erstmals tritt etwas Neues auf: Das Eckfeld, das von den vorhandenen Feldern umschlossen wird, kann auf zwei verschiedene Weisen mit diesen verbunden werden. Trotzdem darf aber nur eine Verbindung hergestellt werden. Außer der Wahl des Randfeldes ist jetzt also auch noch die Wahl der Verbindungen nötig. Dieser Schritt kann so bereits in acht verschiedenen Varianten ausgeführt werden. Klar ist, daß auch hier einige neue Randfelder entstehen.

Schritt 5:

Sie werden sicher schon vermuten, daß dieses Verfahren solange fortgesetzt wird, bis die gesamte Fläche des endgültigen Labyrinths ausgefüllt ist, also solange, bis keine neuen Randfelder mehr dazukommen und die schon vorhandenen aufgebraucht sind. Und genauso verhält es sich auch tatsächlich. Wenn das Labyrinth etwa aus sechs Feldern besteht, werden schon erste Andeutungen sichtbar, daß es sich bei der entstehenden Figur um ein Labyrinth handeln könnte (Bild 4).



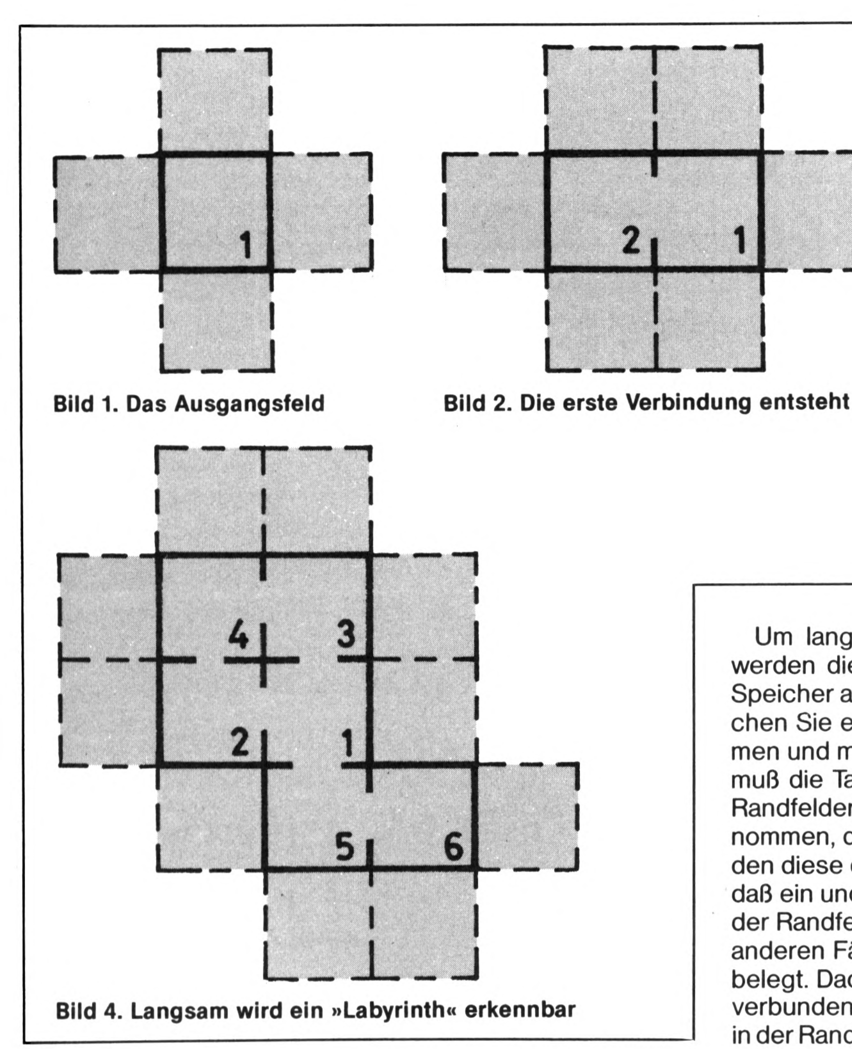


Bild 1. Das Ausgangsfeld

Bild 2. Die erste Verbindung entsteht

Bild 4. Langsam wird ein »Labyrinth« erkennbar

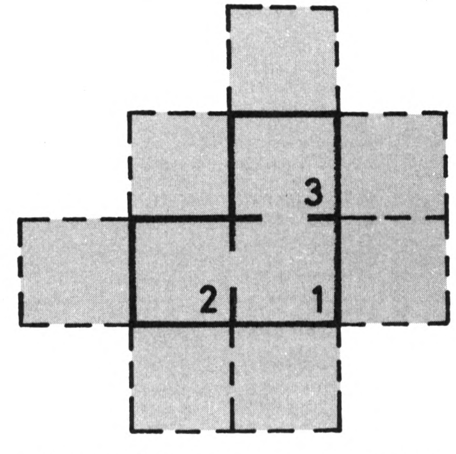


Bild 3. Mit jedem neuen Feld steigt die Zahl der »Randfelder«

Dieses Verfahren erfüllt alle vorausgesetzten Bedingungen: Die ersten beiden Felder sind sicher miteinander verbunden. Ein neu hinzukommendes Feld wird wiederum mit einem dieser Felder verbunden. Auf irgendeine Weise kann so jedes schon vorhandene Feld von neuen Feldern aus erreicht werden.

Jedes Feld ist also mit jedem anderen verbunden. Start- und Endpunkt des Marsches durch das Labyrinth können frei gewählt werden – zum Beispiel nach optischen Gesichtspunkten.

Der Weg zwischen Start- und Endpunkt ist auch eindeutig: Jeder neu dazugekommene Punkt ist durch genau ein »Tor« mit einem Feld des schon vorher bestehenden Teillabyrinths verbunden. Der einzige Weg läuft vom Startpunkt über das erste Feld zum Zielpunkt. Es könnte sein, daß sich die beiden Teilwege kurz vor dem ältesten Feld des Labyrinths treffen. Der restliche Weg wird dann sowohl in der einen, als auch in der anderen Richtung begangen. Wenn Sie diesen Teil weglassen, erhalten Sie wirklich einen einzigen Weg, der zwei beliebige Punkte des Labyrinthes verbindet.

Im Programm (Listing 1) wird für jedes Feld des Labyrinthes ein Byte reserviert. Das Bit 3 ist gesetzt, wenn das Feld schon besetzt ist. Das Bit 2 ist gesetzt, wenn schon eine Verbindung mit dem darunterliegenden Feld vorhanden ist und das Bit 1, wenn das Feld mit dem rechten Nachbarfeld verbunden ist. Das Bit 0 gibt an, ob der Weg vom Start zum Ziel durch dieses Feld läuft. Die Bits 4 bis 7 sind beliebig. Im vorhandenen Programm haben diese Bits die Werte 1, wenn das Feld schon besetzt ist und 0, wenn es noch frei ist. Für besetzte Felder ergeben sich damit die Codes in der nebenstehenden Tabelle. Wenn Sie die Bits 4 bis 7 anders nutzen, ändern sich diese Werte entsprechend.

Um lange Suchzeiten nach Randfeldern zu vermeiden, werden die Orte aller Randfelder in Form einer Tabelle im Speicher abgelegt. Zum Besetzen eines neuen Feldes brauchen Sie ein beliebiges Element aus dieser Tabelle zu nehmen und mit den bisherigen Feldern zu verbinden. Natürlich muß die Tabelle danach unter Berücksichtigung der neuen Randfelder aktualisiert werden. Es werden nur Felder aufgenommen, die den Wert 0 enthalten. Nach der Aufnahme werden diese dann mit dem Wert 3 markiert. So vermeidet man, daß ein und dasselbe Feld fälschlich mehrmals in die Tabelle der Randfelder aufgenommen wird. Andererseits gilt in allen anderen Fällen ein mit dem Wert 3 markiertes Feld nicht als belegt. Dadurch kann es nicht fehlerhaft mit anderen Feldern verbunden werden. Ein Feld, das mit dem Wert 3 markiert und in der Randfeldertabelle vermerkt ist, nimmt also eine Zwitterstellung ein: Es ist weder frei noch besetzt.

Für das Einfügen und Entfernen von Elementen aus der Rand-Tabelle gibt es verschiedene Möglichkeiten. Im vorliegenden Programm hat die Tabelle eine Stapel-Struktur: Die zuletzt eingefügten Elemente werden als erste wieder entnommen. Denkbar wäre auch eine »Random-Access«-Struktur: Ein Element kann durch Zufall gesteuert aus einer beliebigen Position der Tabelle entnommen werden. Damit der Verwaltungsaufwand nicht zu sehr anschwillt, sind allerdings zusätzliche Maßnahmen zu ergreifen: Nach der Entnahme wird das letzte Element in die entstandene Lücke kopiert und die Tabellenlänge um den Wert 1 vermindert. Dadurch trifft die nächste Zufallsauswahl niemals ein schon entnommenes Element. Zusätzlich können neue Elemente einfach am Ende eingefügt werden.

Diesen Tabellenaufbau erreichen Sie, indem Sie die Zeile 60430 des Programms durch »60430 z%=INT(RNDx t%+1):p%=s%(z%)+f%:s%(z%)=s%(t%):t%=t%-1:POKE p%,&F8« ersetzen.

Wert	Verbindung nach		Weg durch das Feld
	unten	rechts	
248	nein	nein	nein
249	nein	nein	ja
250	nein	ja	nein
251	nein	ja	ja
252	ja	nein	nein
253	ja	nein	ja
254	ja	ja	nein
255	ja	ja	ja

Tabelle. Sämtliche Codes für besetzte Felder

```
60000 '*********************************  [0AFA]
60010 '*        L A B Y R I N T H      *  [BFBE]
60020 '*          Vers. (29.05.85)     *  [4056]
60030 '* (c) 1985 by Isar-Amper-Soft   *  [AD68]
60040 '*********************************  [6F02]
60050 '                                   [F584]
60060 'Programm Reset                     [1A56]
60070 MEMORY 19572'Reservierung maximal
      95*134 Felder=(95+2)*(134+2)+2 Byt
      es                                  [F1BC]
60080 SYMBOL &F8,3,3,3,3,3,3,255,255      [A304]
60090 SYMBOL &F9,3,3,3,27,27,3,255,255    [29E0]
60100 SYMBOL &FA,0,0,0,0,0,0,255,255      [ADE4]
60110 SYMBOL &FB,0,0,0,24,24,0,255,255    [E3C0]
60120 SYMBOL &FC,3,3,3,3,3,3,3,3          [1C6C]
60130 SYMBOL &FD,3,3,3,27,27,3,3,3        [FB48]
60140 SYMBOL &FE,0,0,0,0,0,0,3,3          [7E50]
60150 SYMBOL &FF,0,0,0,24,24,0,3,3        [242C]
60160 RANDOMIZE TIME                      [6A2A]
60170 f%=HIMEM+3:ymax%=PEEK(f%-2):xmax%=
      PEEK(f%-1):1%=xmax%+2:h%=ymax%+2    [BE6A]
60180 DIM s%(4999),r%(4)                  [3F52]
60190 r%(1)=1:r%(2)=-1%:r%(3)=-1:r%(4)=1
      %                                   [5A14]
60200 RESTORE 60260                       [91B4]
60210 DIM o%(24,4)'Verschiedene Reihenfo
      lgen der Richtungen 1 bis 4         [1C8A]
60220 FOR i%=1 TO 24:FOR j%=1 TO 4:READ
      o%(i%,j%):NEXT:NEXT                 [60A2]
60230 DIM s1%(7),s2%(7),s3%(7),s4%(7),s5
      %(7)'Drucker-Graphikcharaktere      [38D0]
60240 FOR i%=0 TO 7:READ s1%(i%),s2%(i%)
      ,s3%(i%),s4%(i%),s5%(i%):NEXT       [EC52]
60250 RETURN                              [84FA]
60260 DATA 1,2,3,4,1,2,4,3,1,3,4,2,1,3,2
      ,4,1,4,2,3,1,4,3,2                  [5A10]
60270 DATA 2,3,4,1,2,3,1,4,2,4,1,3,2,4,3
      ,1,2,1,3,4,2,1,4,3                  [9E12]
60280 DATA 3,4,1,2,3,4,2,1,3,1,2,4,3,1,4
      ,2,3,2,4,1,3,2,1,4                  [AA14]
60290 DATA 4,1,2,3,4,1,3,2,4,2,3,1,4,2,1
      ,3,4,3,1,2,4,3,2,1                  [B616]
60300 DATA 1,1,1,1,63,1,13,13,1,63,1,1,1
      ,1,1,1,13,13,1,1                    [A24E]
60310 DATA 0,0,0,0,63,0,12,12,0,63,0,0,0
      ,0,1,0,12,12,0,1                    [7430]
60320 '                                   [5084]
60330 'Labyrinth erstellen                [5DBC]
60340 FOR i%=0 TO 1%-1:POKE f%+i%,3:POKE
       f%+(h%-1)*1%+i%,3:NEXT'Feld loesc
      hen                                 [82EE]
60350 FOR i%=1 TO h%-2                    [8AE0]
60360 POKE f%+i%*1%,3:POKE f%+i%*1%+1%-1
      ,3                                  [1C9A]
60370 FOR j%=f%+i%*1%+1 TO f%+i%*1%+xmax
      %:POKE j%,0:NEXT                    [2056]
60380 NEXT                                [A7C0]
60390 p%=f%+INT(RND*(ymax%-4)+2)*1%+INT(
      RND*(xmax%-4)+2)                    [0F12]
60400 POKE p%,&F8:POKE p%-1,3:POKE p%+1,
      3:POKE p%-1%,3:POKE p%+1%,3         [A8EC]
60410 s%(1)=p%-1-f%:s%(2)=p%+1-f%:s%(3)=
      p%+1%-f%:s%(4)=p%-1%-f%:t%=4        [CAAE]
60420 WHILE t%<>0                         [B3F0]
60430 p%=s%(t%)+f%:t%=t%-1:POKE p%,&F8    [1F38]
60440 z%=INT(RND*24)+1                    [3E0C]
60450 FOR i%=1 TO 4                       [1E72]
60460 d%=r%(o%(z%,i%))+p%:IF PEEK(d%)=0
      THEN t%=t%+1:s%(t%)=d%-f%:POKE d%,
      3                                   [DA0C]
60470 NEXT                                [9EC0]
60480 z%=INT(RND*24)+1                    [AE14]
60490 i%=0:WHILE i%<>4:i%=i%+1:ON o%(z%,
      i%) GOSUB 60530,60540,60550,60560:
      WEND                                [70F6]
60500 WEND                                [7892]
60510 RETURN                              [5FF8]
60520 'Neues Feld mit altem verbinden     [EDB2]
60530 IF PEEK(p%-1 )<8 THEN RETURN ELSE
      POKE p%-1 ,PEEK(p%-1 )OR 2:i%=4:RE
      TURN                                [31BE]
60540 IF PEEK(p%+1%)<8 THEN RETURN ELSE
      POKE p%   ,PEEK(p%   )OR 4:i%=4:RE
      TURN                                [F2C8]
60550 IF PEEK(p%+1 )<8 THEN RETURN ELSE
      POKE p%   ,PEEK(p%   )OR 2:i%=4:RE
      TURN                                [F146]
60560 IF PEEK(p%-1%)<8 THEN RETURN ELSE
      POKE p%-1%,PEEK(p%-1%)OR 4:i%=4:RE
      TURN                                [1E48]
60570 '                                   [5392]
60580 'Weg durch das Labyrinth finden     [C3B8]
60590 start%=ymax%*1%+1:ende%=1%+xmax%'S
      tart+Endfeld links-unten/rechts ob
      en                                  [5458]
60600 p%=start%+f%:POKE p%,PEEK(p%)OR 1:
      r%=1                                [8CFA]
60610 WHILE p%<>ende%+f%                  [C278]
60620 pl%=p%                              [61E2]
60630 ON r% GOSUB 60670,60680,60690,6070
      0                                   [3F80]
60640 IF PEEK(p%)AND 1 THEN POKE pl%,PEE
      K(pl%)AND 254 ELSE POKE p%,PEEK(p%
      )OR 1                               [C3C8]
60650 WEND                                [4B9E]
60660 RETURN                              [F704]
60670 IF PEEK(p%   )AND 2 THEN p%=p%+1 :
      r%=4:RETURN                         [5620]
60680 IF PEEK(p%-1%)AND 4 THEN p%=p%-1%:
      r%=1:RETURN                         [6A60]
60690 IF PEEK(p%-1 )AND 2 THEN p%=p%-1 :
      r%=2:RETURN                         [4F60]
60700 IF PEEK(p%   )AND 4 THEN p%=p%+1%:
      r%=3:RETURN                         [9996]
60710 GOTO 60670                          [FCF4]
60720 '                                   [F88C]
60730 'Labyrinth auf Bildschirm ausgeben
                                          [40B6]
60740 MODE 1                              [242E]
60750 PRINT CHR$(254);:FOR i%=1 TO xmax%
      :PRINT CHR$(250);:NEXT              [FCFC]
60760 IF POS(#0)<>1 THEN PRINT            [D11E]
60770 FOR i%=1 TO ymax%                   [72DC]
60780 PRINT CHR$(252);                    [28F0]
60790 FOR j%=f%+i%*1%+1 TO f%+i%*1%+xmax
      %:PRINT CHR$(PEEK(j%));:NEXT        [B46C]
60800 IF POS(#0)<>1 THEN PRINT            [FC14]
60810 NEXT                                [9EBC]
60820 RETURN                              [E500]
60830 '                                   [1F90]
60840 'Druckerausgabe ohne Weg            [CCFC]
60850 w%=254:GOTO 60890                   [5662]
60860 '                                   [2296]
60870 'Druckerausgabe mit Weg             [B342]
60880 w%=255                              [5336]
60890 'Grundprozedur Druckerausgabe       [3A22]
60900 lo%=(xmax%*5+3)AND 255:hi%=INT((xm
      ax%*5+3)/256):re%=0'Breite des Bil
      des                                 [376C]
60910 IF lo%>127 THEN hi%=hi%+1:re%=256-
      lo%:lo%=0'Korrektur f}r fehlendes
      8.Bit                               [EAAE]
60920 |ESC,27,65,6,27,50,27,56'Graphik v
      orbereiten                          [3814]
60930 |ESC,13,10,27,75,lo%,hi%,0,0,1:FOR
       i%=1 TO xmax%:|ESC,1,1,1,1,1:NEXT
                                          [AD08]
60940 FOR j%=1 TO re%:|ESC,0:NEXT'Oberer
       Rand                               [452E]
60950 FOR i%=1 TO ymax%:|ESC,13,10,27,75
      ,lo%,hi%,0,0,63'Neue Zeile vorbere
      iten                                [39EC]
60960 FOR j%=f%+i%*1%+1 TO f%+i%*1%+xmax
      %'Zeile ausgeben                    [879A]
60970 z%=PEEK(j%)-248 AND w%              [FF02]
60980 |ESC,s1%(z%),s2%(z%),s3%(z%),s4%(z
      %),s5%(z%)                          [5202]
60990 NEXT                                [B5CE]
61000 FOR j%=1 TO re%:|ESC,0:NEXT'Korrek
      tur f}r Fehlendes 8. Bit            [4570]
61010 NEXT                                [CCAE]
61020 |ESC,13,10,27,65,12,27,50,27,57'Vo
      rhergehenden Modus wiedereinstelle
      n                                   [03E2]
61030 RETURN                              [DEF4]
```

Listing 1. Der »Grundstock« für Programme besteht aus mehreren Unterroutinen, die per »GOSUB <Zeile>« aufzurufen sind.

Eine dritte Möglichkeit besteht darin, die Rand-Tabelle in einer »Puffer«-Struktur aufzubauen: Die Elemente, die zuerst eingefügt wurden, werden auch als erste wieder entnommen. Das bedeutet, daß neue Elemente am Ende angefügt und am Anfang entnommen werden. Auch wenn nur wenige Werte gleichzeitig zu merken sind, wandert diese Tabelle zu immer höheren Adreßwerten. Deshalb ist es günstig, die Tabelle als Ringpuffer aufzubauen: Wenn das Tabellenende überschritten werden sollte, beginnt man wieder von neuem beim Element Nummer 0. Für diese Tabellenstruktur ist im Programm etwas mehr zu ändern:

60420 tu%=0:WHILE t%< >tu%

60430 tu%=(tu%+1)AND (tu%<4999):p%=s%(tu%)+f%:POKE p%,&F8

60460 ... THEN t%=(t%+1)AND(t%<4999):s%(t%)=d%-f%:poke d%,3

Sie sollten einige Experimente mit den verschiedenen Programmversionen durchführen und schauen, wie sich das auf die Form der Labyrinthe auswirkt.

Das Labyrinth wird in den Zeilen 60420 bis 60560

```
100 '**********************************     [CED8]
110 '* Befehlserweiterung   ' E S C ' *     [70C0]
120 '*        f}r CPC464/664/6128      *     [640C]
130 '* Vers. 2, (24.06.85, 02.02.86)   *     [9D76]
140 '* (c) 1985 by Isar-Amper-Soft     *     [5820]
150 '**********************************     [F3E2]
160 'Erlaubt es, (Steuer-)yeichen ohne       [582C]
170 'Z{hlung durch die Width-Routine         [A91A]
180 'des Basic-Systems an den Drucker        [80F2]
190 'auszugeben                              [8E34]
200 '                                        [E4B2]
210 '                                        [05B4]
220 'Programm einladen                       [A000]
230 '   (mit Verschiebelader)                [BCBE]
240 SYMBOL AFTER 256                         [97B6]
250 READ n,m:st=HIMEM-n+1:MEMORY st-1        [6D6E]
260 FOR i=0 TO n-1                           [5C88]
270 READ d$:d=VAL("&"+d$):POKE st+i,d        [F67A]
280 NEXT                                     [70F2]
290 FOR i=1 TO m                             [A4D2]
300 READ d$:d=VAL("&"+d$)                    [B446]
310 z=st+PEEK(st+d)+256*PEEK(st+d+1)         [2646]
320 POKE st+d+1,INT(z/256)                   [7508]
330 POKE st+d,z-256*INT(z/256)               [982E]
340 NEXT                                     [77EC]
350 '                                        [05BE]
360 'Initialisierung                         [5420]
370 CALL HIMEM+1                             [8184]
380 '                                        [0AC4]
390 'beliebige eigene Erg{nzungen            [148C]
400 '**********************************     [12DE]
410 OPENOUT"##":MEMORY HIMEM-1:CLOSEOUT      [4324]
420 SYMBOL AFTER 240                         [98A8]
430 '**********************************     [CFE4]
440 '                                        [E4BE]
450 END                                      [B020]
460 '                                        [E2C2]
470 'L{nge  und                              [4AD4]
480 'Anzahl der anzupassenden Adressen       [0B20]
490 DATA 51,6                                [D47E]
500 'Programm (ab Adresse &0000)             [1D84]
510 DATA 01,09,00,21,12,00,C3,D1             [0CBE]
520 DATA BC,0E,00,C3,16,00,45,53             [511A]
530 DATA C3,00,00,00,00,00,A7,C8             [DBDE]
540 DATA 47,DD,E5,E1,87,5F,16,00             [3276]
550 DATA 19,2B,2B,7E,CD,2A,00,10             [5262]
560 DATA F8,C9,CD,30,00,30,FB,C9             [2DA8]
570 DATA CF,F2,87                            [267C]
580 'Adressen der zu {ndernden Adressen      [7682]
590 DATA 0001,0004,0009,000C                 [DB3A]
600 DATA 0025,002B                           [586E]
```

Listing 2. Damit Sie die Früchte Ihrer Arbeit auch in den Händen halten können, muß dieser Befehl aktiviert sein

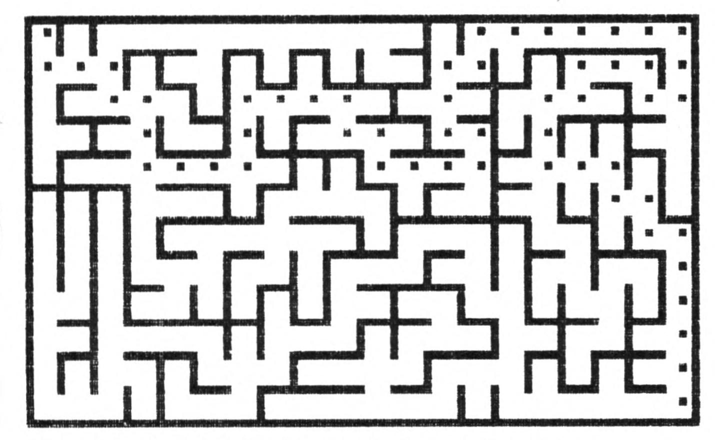
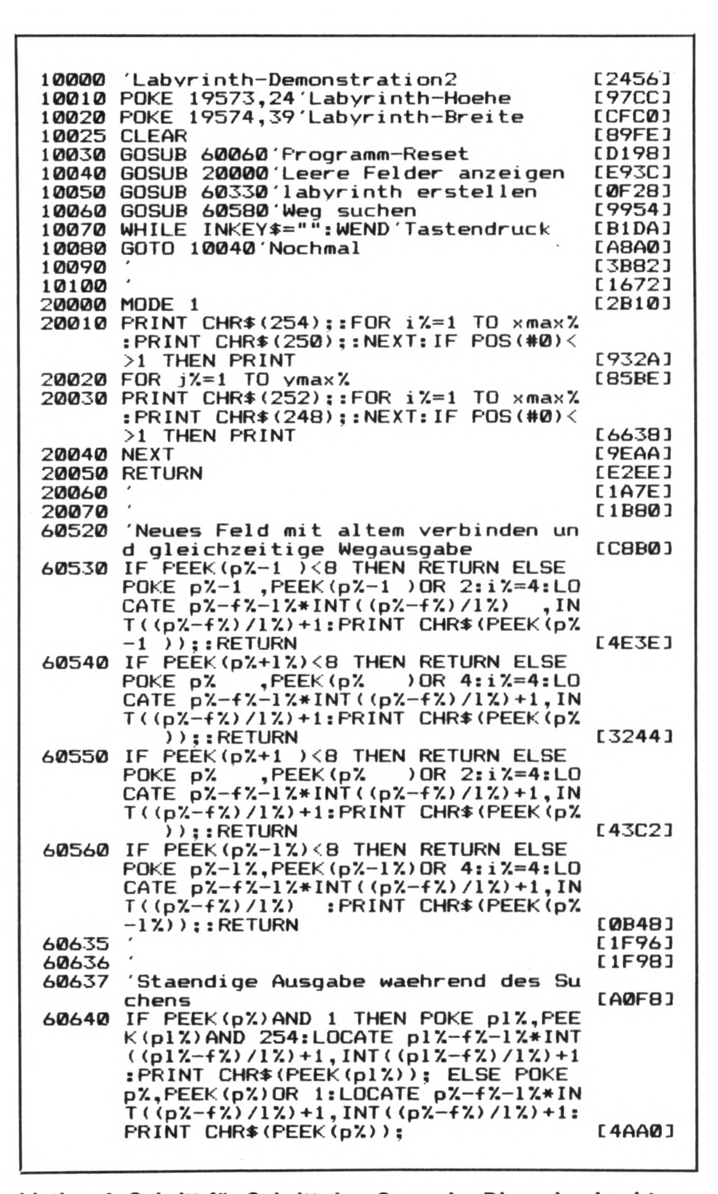


Bild 5. Ein 20 x 12-Beispiel-Labyrinth, erstellt mit dem Programm aus Listing 2

```
10000 'Labyrinth-Demonstration                 [BAF2]
10010 POKE 19573,5 'Labyrinth-Hoehe            [ACAA]
10020 POKE 19574,10 'Labyrinth-Breite          [FBEA]
10030 'Maximale Groe~e:                        [D766]
10040 '95 Felder breit und 134 hoch            [C666]
10050 'Bildschirmausgabe bis 24*39;            [3B96]
10060 'dann nur noch Drucker erlaubt           [3236]
10070 CLEAR                                    [84FE]
10080 GOSUB 60060 'Programm aktivieren         [2126]
10090 GOSUB 60330 'Labyrinth erstellen         [1E30]
10100 GOSUB 60580 'Weg suchen                  [658A]
10110 GOSUB 60730 'Labyrinth anzeigen          [2E30]
10120 GOTO 10090 'Wiederholung                 [A436]
10130 '                                        [3978]
10140 '                                        [127A]
```

Listing 3. So läßt sich der »Labyrinth-Generator« einsetzen



```
10000 'Labyrinth-Demonstration2                [2456]
10010 POKE 19573,24'Labyrinth-Hoehe            [97CC]
10020 POKE 19574,39'Labyrinth-Breite           [CFC0]
10025 CLEAR                                    [89FE]
10030 GOSUB 60060'Programm-Reset               [D198]
10040 GOSUB 20000'Leere Felder anzeigen        [E93C]
10050 GOSUB 60330'labyrinth erstellen          [0F28]
10060 GOSUB 60580'Weg suchen                   [9954]
10070 WHILE INKEY$="":WEND'Tastendruck         [B1DA]
10080 GOTO 10040'Nochmal                       [A8A0]
10090 '                                        [3B82]
10100 '                                        [1672]
20000 MODE 1                                   [2B10]
20010 PRINT CHR$(254);:FOR i%=1 TO xmax%
      :PRINT CHR$(250);:NEXT:IF POS(#0)<
      >1 THEN PRINT                            [932A]
20020 FOR j%=1 TO ymax%                        [85BE]
20030 PRINT CHR$(252);:FOR i%=1 TO xmax%
      :PRINT CHR$(248);:NEXT:IF POS(#0)<
      >1 THEN PRINT                            [6638]
20040 NEXT                                     [9EAA]
20050 RETURN                                   [E2EE]
20060 '                                        [1A7E]
20070 '                                        [1B80]
60520 'Neues Feld mit altem verbinden un
      d gleichzeitige Wegausgabe               [C8B0]
60530 IF PEEK(p%-1 )<8 THEN RETURN ELSE
      POKE p%-1 ,PEEK(p%-1 )OR 2:i%=4:LO
      CATE p%-f%-1%*INT((p%-f%)/1%)  ,IN
      T((p%-f%)/1%)+1:PRINT CHR$(PEEK(p%
      -1 ));:RETURN                            [4E3E]
60540 IF PEEK(p%+1%)<8 THEN RETURN ELSE
      POKE p%   ,PEEK(p%   )OR 4:i%=4:LO
      CATE p%-f%-1%*INT((p%-f%)/1%)+1,IN
      T((p%-f%)/1%)+1:PRINT CHR$(PEEK(p%
      ));:RETURN                               [3244]
60550 IF PEEK(p%+1 )<8 THEN RETURN ELSE
      POKE p%   ,PEEK(p%   )OR 2:i%=4:LO
      CATE p%-f%-1%*INT((p%-f%)/1%)+1,IN
      T((p%-f%)/1%)+1:PRINT CHR$(PEEK(p%
      ));:RETURN                               [43C2]
60560 IF PEEK(p%-1%)<8 THEN RETURN ELSE
      POKE p%-1%,PEEK(p%-1%)OR 4:i%=4:LO
      CATE p%-f%-1%*INT((p%-f%)/1%)+1,IN
      T((p%-f%)/1%)  :PRINT CHR$(PEEK(p%
      -1%));:RETURN                            [0B48]
60635 '                                        [1F96]
60636 '                                        [1F98]
60637 'Staendige Ausgabe waehrend des Su
      chens                                    [A0F8]
60640 IF PEEK(p%)AND 1 THEN POKE pl%,PEE
      K(pl%)AND 254:LOCATE pl%-f%-1%*INT
      ((pl%-f%)/1%)+1,INT((pl%-f%)/1%)+1
      :PRINT CHR$(PEEK(pl%)); ELSE POKE
      p%,PEEK(p%)OR 1:LOCATE p%-f%-1%*IN
      T((p%-f%)/1%)+1,INT((p%-f%)/1%)+1:
      PRINT CHR$(PEEK(p%));                    [4AA0]
```

Listing 4. Schritt für Schritt den Gang der Dinge beobachten

erzeugt. Die Variable t% gibt an, wieviele Randfelder noch in der Tabelle enthalten sind. Die äußere WHILE-Schleife wird solange ausgeführt, bis keine Randfelder mehr vorhanden sind.

In den Zeilen 60450 bis 60470 wird die Tabelle der Randfelder aktualisiert. Die Reihenfolge, in der die vier Nachbarfelder des neuen Feldes abgefragt werden, ist zufällig. Die vier möglichen Richtungen haben die Nummern 1 bis 4. In der Tabelle o% stehen die 24 verschiedenen Reihenfolgen, in der man diese Zahlen aufschreiben kann. Der Zufallszahlengenerator wählt die Reihenfolge mit der Nummer z%. Durch »o%(z%, <Schritt>)« wird festgestellt, welche Richtung als nächste untersucht wird.

Die Zeile 60490 verbindet das neue Element mit einem der alten. Dazu werden die Nachbarfelder solange abgesucht, bis ein schon besetztes Feld gefunden wird. Die Reihenfolge bestimmt wieder der Zufallszahlengenerator und das Variablenfeld o%. Ob eine Verbindung in eine bestimmte Richtung möglich ist, wird in den Unterprogrammen in den Zeilen 60530 bis 60560 festgestellt. Konnte eine Verbindung

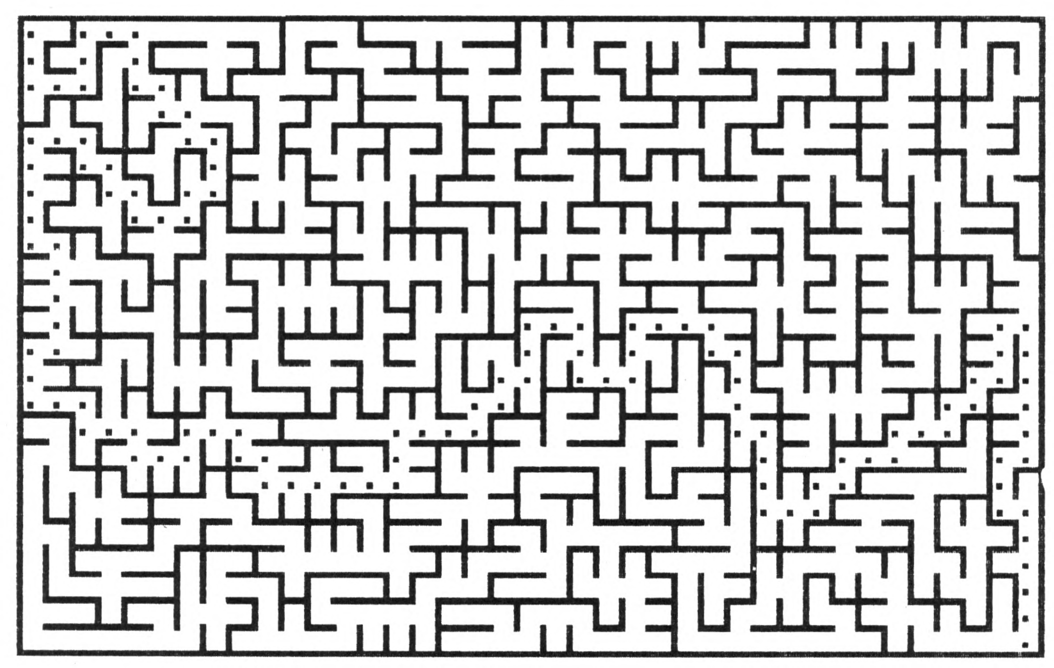


Bild 6. Auch dieses 40 x 24-Labyrinth entstand mit Listing 2

geschaffen werden, wird durch »i% = 4« ein entsprechendes Flag gesetzt.

In den Zeilen 60340 bis 60380 wird das gesamte Labyrinth gelöscht. Der Rand besteht aus Bytes, die den Wert 3 haben. Dadurch sind die Felder außerhalb der Grenzen des Labyrinths für die Aufnahme in die Randfelder-Tabelle gesperrt und das Labyrinth kann nicht ins Uferlose wachsen. Gleichzeitig wird eine Verbindung eines Elementes mit dem Rand vermieden. In den Zeilen 60390 bis 60410 wird das allererste Feld des Labyrinths belegt und dessen vier Nachbarfelder in die Randfelder-Tabelle eingetragen.

Vom Unterprogramm hängt die Labyrinthgröße ab

Das gesamte Labyrinthprogramm ab der Zeile 60000 ist in mehrere Unterprogramme aufgeteilt, die von einem Anwenderprogramm aus aufgerufen werden können.

Das Unterprogramm »Programm-Reset« reserviert Speicherplatz für das Labyrinth, definiert alle Variablen und Sonderzeichen neu und legt die Labyrinthgröße fest: Aus der Speicheradresse 19573 wird der Wert für die Höhe des Labyrinths genommen und aus der Adresse 19574 der Wert für die Breite des Labyrinths. Dorthin werden sie vom Hauptprogramm durch »POKE« gebracht. Es muß jedesmal aufgerufen werden, wenn die Größe des Labyrinths zu verändern ist. Die maximale Größe beträgt 95 Spalten und 134 Zeilen. Diese Werte kommen nicht von ungefähr: Wenn das Labyrinth auf den Drucker ausgegeben wird, nimmt ein 95 x 134-Labyrinth exakt den Platz einer randlosen DIN-A4-Seite ein.

Sie werden sicher schon bemerkt haben, daß die Daten für das Labyrinth nicht in einem Integer-Variablenfeld abgelegt werden, sondern mit Hilfe von »POKE« direkt im Speicher. Dadurch wird der Speicher besser ausgenutzt, so daß es erst möglich wird, so große Labyrinthe zu berechnen: Ein Feld eines Labyrinths belegt nur vier Bit, eine Integerzahl eines Variablenfeldes aber volle 16 Bit. Bei großen Labyrinthen mit 10000 Feldern werden aus diesen 12 Bits Ersparnis schnell viele KByte. Bei der direkten Ablage im Speicher werden pro Feld nur noch acht Bit (inklusive Zeiger für Verbindungen) belegt, also eine große Einsparung. Ein weiterer Vorteil ist, daß ein fertiges Labyrinth durch »SAVE" <Name>.BIN", b,19573, <Länge>« gespeichert werden kann. Bei großen Labyrinthen, bei denen die Rechenzeit leicht zu einer Kaffeepause ausreicht, lohnt sich das schon.

Durch »GOSUB 60580« wird ein Weg von einem vorzugebenden Startpunkt zu einem Zielpunkt gesucht. Das Unterprogramm ab der Zeile 60730 zeigt das Labyrinth auf dem Bildschirm an – auch mit Weg. Diese Routine arbeitet jedoch nur mit Labyrinthen sinnvoll, die maximal 39 x 24 Felder groß sind – das entspricht genau einer Bildschirmseite im Modus 1.

Zwei weitere Routinen erlauben, das berechnete Labyrinth auf den Drucker auszulisten. Dabei kann zwischen der Ausgabe mit oder ohne Lösung gewählt werden.

Den Weg durch ein Labyrinth dieser Sorte – ohne Schleifen – können Sie finden, indem Sie mit der rechten Hand immer an der Wand entlangstreichen und niemals loslassen. Das heißt, Sie folgen immer den rechten Abzweigungen eines Weges. Wenn Sie in eine Sackgasse kommen, gehen Sie an der linken Seite wieder zurück. So kommen Sie zwangsläufig irgendwann zum Ausgang.

Aus jedem Labyrinth der schnellste Weg ins Freie

Genauso geht auch das Programm vor: Die aktuelle Richtung steht in der Variable »r%«. In Zeile 60630 wird das Unterprogramm aufgerufen, das in der am weitesten rechts liegenden Richtung nach einem Weg sucht. Wenn das auf Anhieb nicht glückt, wird die linke Richtung abgefragt und so weiter bis zum Erfolg. Auf seiner Suche hinterläßt das Programm im Bit 0 jedes Raumes eine Markierung. Wenn dieser dann (beim Rückweg) zum zweitenmal betreten wird, kennzeichnet dessen Byte eine Sackgasse. In diesem Fall wird die Markierung wieder gelöscht. Übrig bleibt zum Schluß der Lösungsweg durch das Labyrinth.

In den Variablen start% und ende% muß die Adresse des Start- und Endfeldes relativ zum Labyrinth-Anfang angegeben werden. Zu beachten ist, daß diese Adressen innerhalb des Labyrinths liegen müssen: Sonst sucht das Programm bis in alle Ewigkeit danach.

Die Drucker-Routine arbeitet nur, wenn die Befehlserweiterung »ESC« aktiviert ist. Den Programmtext finden Sie in Listing 2. Wenn Sie als fertige Labyrinthe drucken lassen wol-



len, müsen Sie vor Laden der Labyrinth-Routinen zunächst »ESC« laufen lassen. Andernfalls können Sie darauf natürlich verzichten.

Ein korrekter Druck entsteht nur auf Druckern, bei denen die Grafikausgabe auf folgendem Prinzip beruht: Zuerst übermittelt der Computer eine Steuerzeichenfolge, die dem Drucker mitteilt, daß die darauf folgenden Bytes nicht als normale Zeichen, sondern als Bitmuster zu interpretieren sind. Nach Empfang der festgelegten Zahl von Grafikzeichen schaltet sich der Drucker selbsttätig in den Textmodus zurück. Keinesfalls darf der Grafikmodus mit Hilfe des achten Datenbits (Steuer-Codes größer als 127) einzustellen sein. Diese Bedingungen sind beispielsweise bei Epson-kompatiblen Druckern immer erfüllt.

Das Programm ist an einen Schneider NLQ 401 angepaßt. Für Epson-Drucker und manche dazu kompatible müssen Sie die Bytes <27,50> aus der Zeile 60920 ersatzlos entfernen, da sonst der Zeilenvorschub nicht richtig funktioniert.

Aus einem Bildschirm-Byte werden beim Druck fünf Grafikzeichen aus einer Tabelle. Die Zeilenlänge wird in ein nieder- und ein höherwertiges Byte geteilt, bevor der Computer sie übermittelt. Bei größeren Breiten kann es aber vorkommen, daß das niederwertige Byte einen Wert größer 128 erhält. In diesen Fällen rundet das Programm die Zeilenlänge auf einen durch 256 teilbaren Wert auf und füllt die überzähligen Bytes mit Nullen. Der beschriebene Effekt tritt zum Beispiel immer bei einer Breite von über 384 Spalten auf. Die meisten Drucker tolerieren den Versuch, 512 Spalten drucken zu wollen, obwohl sie eigentlich nur derer 480 darstellen können.

Zeile 60920 stellt die Zeilenhöhe auf sechs Pixel ein und schaltet die Papiermangel-Anzeige ab, damit Sie auch DIN-A4-Einzelblätter benutzen können. Zeile 61020 stellt den Drucker wieder auf sein Standardwerte zurück. Die Bytes <27,75,10%,hi%> in den Zeilen 60930 und 60950 schalten in den Grafikmodus.

Die Listings 3 und 4 zeigen Anwendungen der Labyrinth-Routinen, Bild 5 und 6 fertige Labyrinthe. Sie müssen jeweils vor dem Test mit Hilfe des Basic-Befehls »MERGE« zum Hauptprogramm dazugeladen werden. Beim zweiten Demo-Programm darf die Reihenfolge auf keinen Fall umgedreht werden!!!

Demonstration 1 zeigt eine einfache Anwendung: Der Bildschirm bleibt zunächst dunkel, bis das fertige Labyrinth auf einen Schlag gezeigt wird.

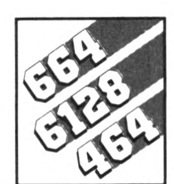
Die Demonstration 2 ist etwas ganz besonderes: Jetzt werden die Labyrinth-Routinen nicht nur aufgerufen, sondern es werden ab Zeile 60000 einige Zeilen verändert. Das hat zur Folge, daß Sie jetzt dem Labyrinth-Programm direkt bei der Arbeit zuschauen können: Alle Felder-Besetzungen werden sofort angezeigt, so daß sichtbar wird, wie sich die Wege erst nach und nach durch ein anfangs unberührtes Gitternetz bahnen.

Während sich bei der »Stapel«-Methode erst ein einzelner Weg durch den Bildschirm schlängelt, und erst nach und nach Seitenäste wachsen, bildet bei der »Zufalls«-Methode der schon berechnete Teil des Labyrinths einen kompakten Block mit hervorzüngelnden Ausläufern, der sich ausbreitet wie ein Flächenbrand.

Mit der »Puffer«-Methode entsteht das Labyrinth trotz Zufall äußerst regelmäßig (in der Praxis weniger geeignet).

(Helmut Tischer/ja)

Für welchen von den CPCs?



Es ist schon ein Problem, Programme so zu schreiben, daß sie auf allen Schneider-Computern vom 464 bis zum 6128 laufen. Aber auch diese Klippe läßt sich elegant umschiffen.

Da es Fälle gibt, in denen sich Inkompatibilität nicht vermeiden läßt, sollte man Lösungen finden, die auf je einem der drei Computer funktionieren.

Dazu muß man vom Programm aus feststellen können, auf welchem Computer das Programm gerade abgearbeitet wird. Die Programmierer bei Amstrad haben dafür sogar eine ROM-Routine vorgesehen: KL-PROBE-ROM an der Adresse B915 hex. Ohne ein kleines Maschinencode-Programm läßt sich diese Routine aber nicht erreichen.

Wesentlich einfacher ist es, den Speicheranfang auszulesen. Einen auf allen drei Computern verschiedenen Wert liefert die Adresse 6. Das folgende kurze Basic-Programm zeigt, wie sich der Computertyp anhand dieser Adresse feststellen läßt:

```
»IF PEEK(6)=128 THEN PRINT "Ein CPC-464!"«
»IF PEEK(6)=&7B THEN PRINT "Ein CPC-664!"«
»IF PEEK(6)=&91 THEN PRINT "Ein CPC-6128!"«
```

(Martin Kotulla/ja)

Fehler im Basic!



Wie bei praktisch jeder komplexen Software-Entwicklung haben sich auch ins Locomotive-Basic kleine Fehler eingeschlichen. Sie lassen sich jedoch umgehen.

Völlig unsinnig erscheint der folgende Befehl:
»INPUT #8,"Eingabe: ",a$«

Er bedeutet, daß der Computer über die Centronics-Schnittstelle Daten einlesen soll, obwohl dies hardwaremäßig keinesfalls möglich ist! Wenn Sie den Befehl einmal ausprobieren, werden Sie aber merken, daß ihn der Computer ohne Fehlermeldung »schluckt«. Er bewirkt, daß der angegebene Text (hier »Eingabe:«) an den Drucker geschickt und die Eingabe von a$ über die Tastatur erwartet wird.

Einen wirklichen Programmierfehler haben sich aber die Entwickler bei den Windows des CPC 464 geleistet:

```
10 MODE 1:WINDOW #5,10,20,10,35
20 LINE INPUT #5,"Eingabe. ",a$
```

"Eingabe : " erscheint im Standard-Fenster 0! Während die Eingabe im Window 5 erfolgt. Abhilfe:

```
10 MODE 1:WINDOW #5,10,20,10,35
20 PRINT #5,"Eingabe: ";:LINE INPUT #5,a$
```

(Martin Kotulla/ja)

Neue Editorfunktionen



Die Schneider-Computer besitzen einen etwas gewöhnungsbedürftigen, aber ausgezeichneten Basic-Editor. Doch nichts ist so gut, als daß man es nicht noch verbessern könnte.

Beim CPC 464 und CPC 664 ist zwar der Cursorblock gut vom Haupt-Tastenfeld abgesetzt, liegt aber sehr weit oben. Da wo die Cursortasten eigentlich besser aufgehoben wären, befindet sich der Zehnerblock. Das führt dazu, daß Sie bei Steuerung des Copy-Cursors entweder beide Hände zu Hilfe nehmen müssen oder sich fast die Fingergelenke verrenken. Besitzer eines CPC 6128 sind zumindest in dieser Beziehung im Vorteil, wenn auch die dortige Tastenanordnung ansonsten einigen Anlaß zur Kritik gibt. Die Idee: Weil der Zehnerblock sowieso kaum benötigt wird, kann man ihn als Cursorfeld gebrauchen. Das Schneider-Basic bietet zur Herstellung der neuen Funktion die Befehle KEY und KEY DEF.

Das Listing enthält ein Programm, um den Zehnerblock folgendermaßen zu belegen:

– Auf den Tasten F2, F4, F6 und F8 liegen die normalen Cursor-Richtungen wie auch im Cursorblock. Mit Shift und diesen Zifferntasten spricht man den Copy-Cursor an. »Control-F4« bewegt den Cursor an den Anfang der Bildschirmzeile, »Control-F6« an deren Ende. »Control-F8« setzt den Cursor an den Beginn der Eingabezeile, »Control-F2« an deren Ende. Natürlich können auch die normalen Cursortasten weiterhin verwendet werden.

Die Copy-Taste wird erweitert: Mit »Shift-Copy« übernimmt der Copy-Cursor die nächsten fünf, mit »Control-Copy« die nächsten zehn Zeichen. Gleichwertig mit Copy ist auch die Taste F5.

– Die Tasten F1, F3, F7 und F9 bieten eine völlig neue Funktion, nämlich die diagonale Bewegung des Cursors. Die Belegung der Tasten entspricht dabei der logischen Bewegungsrichtung, also bewegt F1 den Cursor nach links unten, F3 nach rechts unten und so weiter.

– Die Taste F0 simuliert eine Tabulator-Funktion: »F0« setzt den normalen Cursor fünf Zeichen nach rechts, »Shift-F0« und »Control-F0« fünf Zeichen nach links.

– Die Punkt-Taste im Zehnerblock ist ein Tabulator für den Copy-Cursor: Sie bewegt ihn um fünf Zeichen nach rechts, während »Shift-Punkt« und »Control-Punkt« dasselbe nach links bewirken.

– Die Funktion der CLR-Taste wird erweitert: Neben der normalen Funktion, ein Zeichen zu löschen, gibt es jetzt »Shift-CLR« zum Löschen von fünf und »Control-CLR« zum Löschen von zehn Zeichen.

– Die DEL-Taste wird ebenfalls verbessert: Während »DEL« weiterhin ein Zeichen löscht, entfernt »Shift-DEL« wieder fünf und »Control-DEL« zehn Zeichen vom Bildschirm.

Mit diesen Änderungen ist der Editor wesentlich schneller zu bedienen, zumal alle Tasten mit einer automatischen Wiederholungsfunktion ausgestattet sind.

(Martin Kotulla/ja)

```
100 ' ********************************** [58C4]
110 ' *                                * [E35A]
120 ' * Zusaetzliche Editorfunktionen * [85A2]
130 ' *                                * [E15E]
140 ' ********************************** [D4CC]
150 '                                     [E1BA]
160 CALL &BB00 ' Tastatur zuruecksetzen [1AFC]
170 ' --- Normale Cursortasten: 2-4-6-8 [1AAC]
180 KEY DEF 20,1,&F2,&F6,&FA          [95C2]
190 KEY DEF 4,1,&F3,&F7,&FB           [0E6E]
200 KEY DEF 11,1,&F0,&F4,&F8          [439A]
210 KEY DEF 14,1,&F1,&F5,&F9          [67A8]
220 ' --- Diagonale Cursortasten      [9EA6]
230 KEY DEF 13,1,141,141,141          [6D68]
240 KEY 141,CHR$(&F1)+CHR$(&F2)       [5E16]
250 KEY DEF 5,1,142,142,142           [6E14]
260 KEY 142,CHR$(&F1)+CHR$(&F3)       [3F1E]
270 KEY DEF 10,1,143,143,143          [9976]
280 KEY 143,CHR$(&F0)+CHR$(&F2)       [0320]
290 KEY DEF 3,1,144,144,144           [6E24]
300 KEY 144,CHR$(&F0)+CHR$(&F3)       [D816]
310 ' --- 5 = Copy-Cursor             [8634]
320 KEY DEF 12,1,145,150,151          [F870]
330 KEY 145,CHR$(&E0)                 [99E4]
340 KEY 150,STRING$(5,&E0)            [E394]
350 KEY 151,STRING$(10,&E0)           [5EF0]
360 ' --- 0 = Tab, .=Copy-Cursor-Tab  [5858]
370 KEY DEF 15,1,146,147,147          [7098]
380 KEY 146,STRING$(5,&F3)            [A8AE]
390 KEY 147,STRING$(5,&F2)            [4CB0]
400 KEY DEF 7,1,148,149,149           [973A]
410 KEY 148,STRING$(5,&F7)            [15AE]
420 KEY 149,STRING$(5,&F6)            [E1B0]
430 ' --- CLR-Taste                   [5768]
440 KEY DEF 16,1,&10,152,153          [4C60]
450 KEY 152,STRING$(5,&10)            [7C74]
460 KEY 153,STRING$(10,&10)           [4BD0]
470 ' --- DEL-Taste                   [CC58]
480 KEY DEF 79,1,&7F,154,155          [EEBA]
490 KEY 154,STRING$(5,&7F)            [40B8]
500 KEY 155,STRING$(10,&7F)           [1002]
510 ' --- Original-COPY = Taste 5     [7428]
520 KEY DEF 9,1,145,150,151           [8720]
```

Listing. Eine neue komfortablere Cursor-Steuerung erleichert die Eingabe





Wirklich Zufall?



Basic-Programmierer, die Zufallszahlen benötigen, können den DANOMIZE-Befehl und die RND-Funktion verwenden. Doch was machen Maschinensprache-Programmierer?

Wie gesagt, ist es im Maschinen-Programm nicht so ganz einfach, Zufallszahlen zu erhalten. Folgende Möglichkeiten stellen sich dar.

- Das Maschinenprogramm ruft die ROM-Routinen des Betriebssystems auf, in denen Zufallszahlen erzeugt werden. Leider entfällt dabei aber jegliche Portabilität des Programms auf andere Z80-Computer, ja nicht einmal auf andere Modelle der Schneider-CPC-Reihe!
- Das Maschinenprogramm liest den internen Timer aus (Basic-Befehl »PRINT TIME«). Nachteil: Bei Verwendung der vorgesehenen ROM-Routine KL-TIME-PLEASE auf Adresse BD0D hex werden DE- und HL-Register zerstört. Diese auf dem Stack zu sichern, kostet viel Rechenzeit. Man kann die Speicherstellen des Timers auch direkt auslesen, verliert dann aber wieder die Übertragbarkeit auf andere Schneider-Computer.
- Man bedient sich des Refresh-Registers (R-Register) des Z80-Prozessors. Erfreulicherweise hat diese Methode keinen Haken.

Das R-Register enthält acht Bit, von denen sieben nach jedem Befehlsholzyklus der CPU inkrementiert werden. Das achte Bit bleibt unverändert, also auf 0 oder 1. Da der Zeitpunkt des Zugriffs wirklich zufällig ist, läßt sich das Register als Zufallszahlen-Generator verwenden.

Das kurze Maschinenprogramm liest mit »LD A,R« das Refresh-Register aus und stellt durch »AND &7F« sicher, daß der Wert zwischen 0 und 127 liegt. Dann speichert es die Zahl in der Adresse 160 hex, von wo aus man sie mit dem Basic-Befehl »PRINT PEEK(&160)« auslesen kann.

Zur Demonstration dient das Beispiel-Listing, das 10000 Zufallszahlen erzeugt und dann ihre Verteilung ausgibt. Sie müssen etwa 75 Sekunden auf das Resultat warten.

Der amerikanische ANSI-Standard für Zufallszahlengeneratoren von 1968 schreibt vor, daß die Streuung der Zahlen nicht größer als 3 Prozent sein darf. Wenn Sie die Bildschirmausgabe betrachten, werden Sie erkennen, daß dieser Zufallsgenerator dem Standard nicht entspricht. Für normale Zwecke sind die Zahlen aber brauchbar.

(Martin Kotulla/ja)



```
100 ' ********************************** [58C4]
110 ' *                                * [E35A]
120 ' * RANDOM-Zahlen aus dem Refresh * [6E18]
130 ' *                                * [E15E]
140 ' ********************************** [D4CC]
150 '                                     [E1BA]
160 MEMORY &9FFF                          [9CC2]
170 FOR i=&A000 TO &A007:READ a:POKE i,a
    :NEXT i                               [4C2A]
180 DATA &ED,&5F,&E6,&7F,&32,&60,&01,&C9
                                          [07F6]
190 DIM random(127)                       [BA40]
200 FOR i=1 TO 10000:CALL &A000           [9FA2]
210 a=PEEK(&160)                          [C108]
220 random(a)=random(a)+1                 [6C66]
230 NEXT i                                [ECFA]
240 FOR i=0 TO 127                        [7120]
250 PRINT USING "Zahl ### war #### mal v
    orhanden";i,random(i)                 [408C]
260 NEXT i                                [2700]
```

Listing. Zufallszahlenerzeugung in Maschinensprache

ESCAPE-Taste blockiert



In Basic-Programmen ist es häufig wünschenswert, die ESCAPE-Taste außer Gefecht zu setzen. Mehrere unterschiedliche Wege führen zu diesem Ziel.

Jeder Basic-Programmierer kennt wohl den ON-BREAK-GOSUB-Befehl. Mit ihm läßt sich ein Programmabbruch verhindern:

10 ON BREAK GOSUB 2000
20 PRINT "ESC ist blockiert!":GOTO 20
2000 RETURN

Wenn – wie hier in Zeile 2000 – in der Unterbrechungsroutine nur ein RETURN-Befehl steht und keine anderen Aktionen durchgeführt werden sollen, läßt sich das erheblich einfacher programmieren:

10 KEY DEF 66,0,0,0,0
20 PRINT "Esc ist blockiert!":GOTO 20

Durch den KEY-DEF-Befehl wird auf die ESCAPE-Taste der ASCII-Code 0 gelegt, und zwar in allen drei Tastaturebenen (normal, mit Shift und mit Control). Diese Programmierung hat den Vorteil, daß auch bei INPUT und LINE INPUT das Drükken der ESCAPE-Taste nicht zum Programmabbruch führt.

Beide Befehle haben aber auch den Nachteil, daß man durch einmaliges Drücken der ESCAPE-Taste das Programm zumindest anhalten, wenn auch nicht abbrechen kann. Besonders in Spielen ist es natürlich recht ärgerlich, wenn der Spieler in aller Ruhe die Situation analysieren kann! Abhilfe schafft eine ROM-Routine:

10 CALL &BB48
20 PRINT »Programm läßt sich nicht anhalten!«:GOTO 20

Dieser Call-Aufruf erlaubt aber bei INPUT doch wieder den Programmabbruch. Ist das unerwünscht, kann man mit dem KEYDEF-Befehl kombinieren.

10 CALL &BB48:KEY DEF 66,0,0,0,0
20 INPUT a$

Oft ist es auch wichtig, die Tastenkombination CTRL-SHIFT-ESC, die einen Reset (Zurücksetzen des Computers) hervorruft, zu blockieren:

10 POKE &BDEE,&C9
20 PRINT "Auch RESET ist blockiert!":GOTO 20

In der Adresse BDEE hex steht normalerweise ein Sprungvektor auf die Betriebssystem-Routine KM-TEST-BREAK. Sie prüft, ob ein Abbruchereignis oder ein Reset ansteht. Durch den POKE-Befehl kann der Computer die Routine nicht mehr aufrufen, und die Tastenkombination hat keinerlei sichtbare Wirkung mehr.

(Martin Kotulla/ja)

Trickreiche Steuerzeichen



Mit den Bildschirm-Steuerzeichen der Schneider CPCs kann man erheblich mehr anfangen, als aus dem ansonsten recht brauchbaren Benutzer-Handbuch deutlich wird.

Zwei Befehle, die eigentlich immer wieder benötigt werden – der Anwendungsfall Textverarbeitung oder Tabellenkalkulation – fehlen leider im Schneider-Basic: Das Einfügen und Löschen einer Bildschirmzeile. Mit den Steuerzeichen läßt sich das aber recht einfach erledigen. Das Listing zeigt, wie zwei solcher Steuerzeichen-Strings zusammengesetzt werden:

Die Funktion »FN ins$(zeile)« fügt in der angegebenen Zeile eine Leerzeile auf dem Bildschirm ein und schiebt den Rest des Bildschirms nach unten.

Die Funktion »FN del$(zeile)« löscht die angegebene Zeile vom Bildschirm und zieht den Rest des Bildschirms nach oben.

Das läßt sich ganz einfach durch Verwendung von Window-Befehlen realisieren. Blättern Sie doch einmal im Benutzer-Handbuch Ihres Computers bis zum Kapitel mit den Bildschirm-Steuerzeichen und versuchen Sie, die Definitionen nachzuvollziehen!

In diesem Kapitel findet sich aber auch ein ziemlich schwerer Fehler: Das Steuerzeichen 26, mit dem sich ein Bildschirm-Fenster setzen läßt, erwartet nicht etwa alle Angaben im Wertebereich 1 bis 80, sondern die Zeilenangabe von 0 bis 24 und die Spaltenangabe von 0 bis 79!

Das ist wichtig zu beachten, da sonst Ihre Berechnungen niemals mit dem tatsächlichen Ergebnis auf dem Bildschirm übereinstimmen.

(Martin Kotulla/ja)

```
100 ' ********************************        [AD74]
110 ' *                              *        [345A]
120 ' * FNins$(row) & FNdel$(row)    *        [6F6A]
130 ' *                              *        [2A5E]
140 ' ********************************        [797C]
150 '                                         [E1BA]
160 INPUT "Welcher MODE? ",mo:MODE mo         [B9F8]
170 FOR i=1 TO 24:PRINT "***************"
    ;i:NEXT i                                 [F1F8]
180 DEF FN ins$(x)=CHR$(26)+CHR$(0)+CHR$
    (79)+CHR$(x)+CHR$(24)+CHR$(11)+CHR$(
    11)+CHR$(26)+CHR$(0)+CHR$(79)+CHR$(0
    )+CHR$(24)                                [AE92]
190 DEF FN del$(x)=CHR$(26)+CHR$(0)+CHR$
    (79)+CHR$(x)+CHR$(24)+CHR$(31)+CHR$(
    255)+CHR$(25)+CHR$(10)+CHR$(26)+CHR$
    (0)+CHR$(79)+CHR$(0)+CHR$(24)             [D666]
200 FOR i=1 TO 10:PRINT FN ins$(10);:NEX
    T i                                       [E298]
210 FOR i=1 TO 10:PRINT FN del$(10);:NEX
    T i                                       [2170]
220 GOTO 200                                  [ED3E]
```

Listing. Löschen und Einfügen von Textzeilen auf dem Bildschirm

---

Besseres Basic



Es gibt eine ganze Reihe kleiner Tricks, die das Programmieren in Schneider-Basic erheblich vereinfachen, obwohl sie nicht im Handbuch stehen.

Die STRING$-Funktion:

Neben der üblichen Form »STRING$ (45,″*″)« gibt es noch eine andere Schreibweise mit dem gleichen Effekt »STRING$(45,42)«. Statt des Zeichens kann also auch dessen ASCII-Code angegeben werden, was grundsätzlich ein Byte spart. Zeichen, die über die Tastatur nicht eingegeben werden können (zum Beispiel »STRING$(50,CHR$(255))«), sparen in der kürzeren Form »STRING$(50,255)« sogar drei Byte.

Der NEXT-Befehl

Eine Programmzeile mit zwei Next-Befehlen direkt hintereinander kann verkürzt werden:

100 FOR i=1 TO 10:FOR j=1 TO 20:NEXT j:NEXT i

Kürzer:

100 FOR i=1 TO 10:FOR j=1 TO 20:NEXT j,i

Noch kürzer, aber wesentlich unübersichtlicher geht es so:

100 FOR i=1 TO 10:FOR j=1 TO 20:NEXT:NEXT

Der RSX-Strich:

Der ASCII-Code 124, der als erstes Zeichen von RSX-Befehlen verwendet wird, darf in REM-Zeilen nicht vorkommen. Aufgrund eines Betriebssystemfehlers würde der Computer zum Beispiel im folgenden Programm die Zeile 110 gar nicht beachten, sondern einfach wieder »READY« melden:

100 ' Llschbefehl

110 PRINT ″Wird nicht beachtet!″:GOTO 110

Ärgerlich ist das, weil der Strich im deutschen Zeichensatz das kleine ″ö″ ist und deshalb durchaus auch in REM-Zeilen gebraucht werden könnte. So ist in Zeile 100 der Llschbefehl im Deutschen ein ″Löschbefehl″. Abhilfe: Umschließen des Textes mit Anführungszeichen.

100 ' ″Llschbefehl″

110 PRINT ″So klappt es!″:GOTO 110

(Martin Kotulla/ja)

Das Betriebsystem CP/M wird zunehmend populärer. Deshalb wenden wir uns mit einer dringenden Bitte an alle CP/M-Programmierer!

Haben Sie Programme unter CP/M geschrieben – und finden Sie, daß auch andere davon profitieren sollten? Dann schicken Sie Ihre Programme an:

Redaktion
Happy-Computer
z.Hd. Herrn Hagedorn
Hans-Pinsel-Straße 2
8013 Haar bei München



Grafik kompakt



Hochauflösende Grafik auf dem Schneider ist eine feine Sache. Wenn man aber den Kassettenrecorder als Speichermedium benutzen muß, verdirbt die mit dem Laden verbundene Wartezeit den ganzen Spaß. Das muß nicht sein, wie ein kleines Maschinencode-Programm beweist.

Grafik ist in. Speziell wenn man Text und Bilder kombinieren kann und dies auch noch durch Basic unterstützt wird, dann macht es richtig Spaß, gute Grafiken zu entwerfen. Ein Problem, das sich dabei immer wieder stellt, ist die Wartezeit beim Speichern. Denn Bilder sind ja meist nicht als Selbstzweck gedacht, sondern sie sollen in andere Programme integriert werden. Und beim Kassettenrecorder fangen da die Probleme an. Was nützen beispielsweise die schönsten Bilder in einem Adventure, wenn man aufgrund der Ladezeit völlig aus dem Spielrhythmus gebracht wird. Nach über drei Minuten (bei 1000 Baud Übertragungsrate) macht auch die schönste Grafik den Ärger nicht mehr wett. Nun kann man zwar mit zwei Grafikspeichern arbeiten und es ist mit einigen POKEs auch möglich, den Kassettenrecorder zur schnelleren Arbeit zu zwingen. Dies alles täuscht jedoch nicht darüber hinweg, daß wir immer nur an den Symptomen kurieren. Die eigentliche »Krankheit« ist der Bildschirmspeicher - besser seine Größe. Denn egal, ob wir im Modus 1 oder 2 oder gar mit vielen Farben im Modus 0 arbeiten, immer ist der Grafikspeicher 16 KByte lang. Beim Speichern entspricht das acht Blöcken und eine solche Menge Daten benötigt eben Zeit. Dabei kommt nun noch ein weiterer ärgerlicher Punkt hinzu. Da der Grafik-Speicher beim Schneider etwas umständlich aufgebaut ist, kann man auch nicht nur einen Teil des Bildschirms sichern, wenn beispielsweise in einem Text/Grafik-Adventure nur die obersten zehn Zeilen überhaupt mit Grafik belegt sind. Hier ist der CPC reichlich stur. Es müssen die vollen 16 KByte übertragen werden.

Was kann man tun? Der einzige Ausweg ist der Verzicht auf einen Teil der hochauflösenden Grafik. Oft ist dies aber gar nicht schlimm. Um beispielsweise ein Labyrinth zu zeichnen, in dem sich der stolze Recke zurechtfinden muß, bedarf es oft keiner großartigen Darstellungsmittel. Eine ausgekochte Blockgrafik tut es auch, besonders wenn man sie wie beim Schneider einfach und schnell umdefinieren kann.

Wie kann man nun aber Bilder oder auch Teilbilder mit der Blockgrafik zeichnen? Dazu müssen wir Änderungen im System vornehmen, die jedoch, wie Sie gleich sehen werden, nicht schwer durchzuführen sind. Der Schneider arbeitet ja ohne einen extra Speicher für Blockgrafik. Dies ist aber auch gerade unser Problem. Daraus folgt nämlich, daß der Computer natürlich auch keine Routinen zur Darstellung von Blockgrafik-Bildern bereithält. Wir müssen uns diese also schreiben. Daneben benötigen wir noch ein Entwicklungsprogramm, das uns hilft, die Bilder zu entwerfen. Viel Arbeit also. Lohnt sich das überhaupt? Wir meinen schon.

Bilder aus dem »Baukasten«

Gehen wir einmal davon aus, daß beim Schneider im Modus 1 25 Textzeilen mit maximal je 40 Zeichen zur Verfügung stehen, so ergeben sich genau 1000 abzuspeichernde Symbole pro Bildschirm. Jedes Symbol benötigt dabei ein Byte für seinen Zeichencode und ein weiteres für die zu setzende Farbe. Alles in allem sind dies maximal 2000 Byte, verglichen mit den 16 KByte des Grafikspeichers also nur ein Achtel des Speicherbedarfs bei hochauflösender Grafik. Im Klartext heißt das, daß wir nur ein Achtel der Speicherzeit benötigen - oder bei der Vorratshaltung von Bildschirmen im Speicher acht Bildschirme auf dem Platz von einem parken können.

Als nächstes müssen wir festlegen, was unser Programm eigentlich können soll. Das Blockgrafik-Entwicklungsprogramm soll uns helfen, an einer beliebigen Stelle auf dem Bildschirm mit Hilfe der Tastatur Grafiksymbole in verschiedenen Farben zu setzen oder zu löschen. Die aktuelle Position soll dabei durch einen blinkenden Cursor angezeigt werden. Es muß jederzeit möglich sein, eine neue Bildschirmfarbe zu wählen oder die Tastatur neu zu definieren, um auf alle möglichen Grafiksymbole zurückgreifen zu können.

Daneben soll das Programm Routinen zum Speichern und Laden von Bildern haben. Die Aufzeichnung soll so universell sein, daß die mit dem Programm - wir wollen es »Designer« nennen - entwickelten Bilder möglichst problemlos in andere Programme einzubinden sind.

Mit dieser Beschreibung haben wir - neben der Zieldefinition im engeren Sinn - auch schon eine Reihe von weiteren Unterpunkten angerissen. Auf diese wollen wir nun näher eingehen.

Welchen grundsätzlichen Aufbau wollen wir wählen? Es bieten sich verschiedene an. Wenn das Programm mit relativ wenigen Meldungen auskommt, so reicht ein einziger Arbeitsbildschirm aus. Auf ihm erfolgen alle Ein- und Ausgaben.

Je größer die Anzahl der Funktionen und damit natürlich auch die Anzahl auszugebender Meldungen ist, desto unübersichtlicher ist dieses Verfahren. Schließlich ist es völlig unmöglich, aus beispielsweise 50 oder 60 Funktionen, die gleichzeitig auf dem Bildschirm dargestellt werden, die richtige auszuwählen. Dabei noch die Übersicht zu behalten ist schwierig.

In solchen Fällen hat sich die Menütechnik als das geeignete Verfahren herausgestellt. Dabei wird von einem Hauptmenü in mehrere Untermenüs (beispielsweise Kassetten- und Diskettenoperationen, Suchroutinen, Darstellungen von Texten und so weiter) verzweigt, die dann wieder ihre eigenen spezifischen Funktionen aufweisen.

Man hat damit in einem Unterpunkt immer nur eine spezifische, auf einen Problembereich ausgerichtete Anzahl von Funktionen zur Auswahl. Diese werden dann auf dem Bildschirm dargestellt. Beim nächsten Arbeitsgang werden andere Funktionen zur Verfügung gestellt.

Bei »Designer« (Listing 1) wollen wir eine Kombination aus beiden Verfahren benutzen. Die eigentliche Bilddarstellung erfolgt auf einem einzigen Bildschirm, der nur durch den Benutzer bei der Zeicheneingabe verändert wird.

Daneben kann man durch Druck auf eine Taste ein Auswahlmenü aufrufen, das dann alle nicht laufend benötigten Funktionen, wie das Abspeichern und Sichern von Bildern oder Teilbildern, die Tastaturumdefinition oder die Farbwahl enthält.

Bei dieser Art der Programmausführung treffen wir aber auf ein Problem, das bei fast allen Grafikentwicklungsprogrammen auftritt. Zum einen wollen wir unsere Bilder auf dem Monitor darstellen, zum anderen benötigen wir den gleichen Platz zur Ausgabe der Arbeitsroutinen. Diese überschreiben damit natürlich unsere Grafik. Wir speichern unsere Bilder daher noch ein zweites Mal - in einem separaten Zeichen- und

```
10 '***************************          [88D8]
20 '** Initialisierungsteil **          [C4C4]
30 '***************************          [12DC]
40 BORDER 0                             [D224]
50 MODE 1                               [F1F6]
60 DIM f(15,2)                          [C9F6]
70 f(0,1)=0:f(0,2)=0                    [B448]
80 f(1,1)=24:f(1,2)=24                  [A926]
90 f(2,1)=6:f(2,2)=6                    [DA6C]
100 f(3,1)=2:f(3,2)=2                   [2EB0]
110 f(4,1)=21:f(4,2)=21                 [8A7A]
120 f(5,1)=15:f(5,2)=15                 [938C]
130 f(6,1)=7:f(6,2)=7                   [05D6]
140 f(7,1)=4:f(7,2)=4                   [64D0]
150 f(8,1)=27:f(8,2)=27                 [6AAA]
160 f(9,1)=18:f(9,2)=18                 [9DB0]
170 f(10,1)=8:f(10,2)=8                 [4F8E]
180 f(11,1)=17:f(11,2)=17               [0A54]
190 f(12,1)=9:f(12,2)=9                 [B99E]
200 f(13,1)=11:f(13,2)=11               [0236]
210 f(14,1)=26:f(14,2)=26               [4054]
220 f(15,1)=24:f(15,2)=6                [2EF2]
230 FOR i=0 TO 15                       [21B6]
240 INK i,f(i,1),f(i,2)                 [BAE8]
250 NEXT i                              [44FE]
260 DIM nr(26),z(6,26)                  [F118]
270 '*********************              [C7A4]
280 '** Masch-PRG laden **              [1DDE]
290 '*********************              [A7A8]
300 MEMORY 39999                        [8D86]
310 KEY DEF 24,1,20                     [7DFA]
320 DATA 21,40,9c,11,28,a0,1a,d5,e5,f5,c
    d,90,bb                             [5526]
330 DATA f1,1f,1f,1f,1f,cd,96,bb,e1,e5,7
    e,cd,5d                             [DA32]
340 DATA bb,e1,d1,23,13,3e,a0,bc,20,e1,3
    e,28,bd                             [2CCE]
350 DATA 20,dc,c9,x                     [1266]
360 i=42010                             [96AC]
370 READ a$:IF a$<>"x" THEN POKE i,VAL("
    &"+a$):i=i+1:GOTO 370               [A524]
380 pe=1:pa=0                           [62EC]
390 MODE 1:PEN 2:PAPER 0                [E346]
400 '*******************************    [6B8E]
410 '** Initialisierungsspruenge **     [78A0]
420 '*******************************    [5992]
430 GOSUB 1750:GOSUB 470:GOSUB 630:GOSUB
     1440:GOSUB 1080:GOTO 800           [61FA]
440 '*********************              [35A2]
450 '** Anfangsabfragen **              [C578]
460 '*********************              [15A6]
470 PRINT:PRINT" Speicher loeschen"     [E32E]
480 FOR i=40000 TO 41000:POKE i,32:NEXT
    i                                   [6474]
490 FOR i=41000 TO 42000:POKE i,0:NEXT i
                                        [9210]
500 CLS                                 [792E]
510 LOCATE 10,3:PRINT"D E S I G N E R{2
    SPACE}4 6 4":PEN 1                  [842C]
520 PRINT:PRINT:PRINT                   [FAA4]
530 PRINT" In welchem Modus wollen Sie a
    rbeiten ?"                          [EF90]
540 PRINT                               [D98C]
550 PRINT"{3 SPACE}40-Zeichen pro Zeile=
    MODE 1{3 SPACE}(1)"                 [D5C2]
560 PRINT"{3 SPACE}20-Zeichen pro Zeile=
    MODE 0{3 SPACE}(0)"                 [29BC]
570 z$=INKEY$:IF z$<>"1" AND z$<>"0" THE
    N 570 ELSE MODE VAL(z$):mz=20+20*VAL
    (z$)                                [C948]
580 IF z$="1" AND pe MOD 4=pa MOD 4 THEN
     pe=MIN(pe,pa)+1                    [057C]
590 RETURN                              [183C]
600 '**************                     [9A52]
610 '** Farbwahl **                     [F6DA]
620 '**************                     [2C56]
630 IF mz=20 THEN j=15 ELSE j=3         [A25A]
640 MODE (mz-20)/20                     [49AE]
650 PRINT"Farbdefinition:":PRINT:PRINT C
    HR$(24):PRINT"Farbe 0"+CHR$(24)     [C0DC]
660 FOR i=1 TO j:PEN i:PRINT"Farbe";i:NE
    XT i                                [843C]
670 PRINT                               [BA94]
680 PEN 1                               [39E4]
690 PRINT"Farbe wechseln j/n?"          [779E]
700 z$=LOWER$(INKEY$):IF z$="n" THEN 760
     ELSE IF z$<>"j" THEN 700           [0B34]
710 INPUT"Welches Farbregister";reg     [5DA0]
720 INPUT"Erste{2 SPACE}Farbe";f(reg,1) [117A]
730 INPUT"Zweite Farbe";f(reg,2)        [2F28]
740 INK reg,f(reg,1),f(reg,2)           [74F0]
750 CLS:GOTO 650                        [E598]
760 RETURN                              [B03A]
770 '***********************            [6C56]
780 '** Joystick-Schleife **            [3FDC]
790 '***********************            [A45A]
800 z$=INKEY$:IF z$=CHR$(240) THEN y=y-1
                                        [F6BE]
810 IF z$=CHR$(241) THEN y=y+1          [9A4C]
820 IF z$=CHR$(242) THEN x=x-1:IF x<1 TH
    EN y=y-1:x=mz                       [1B90]
830 IF z$=CHR$(243) THEN x=x+1:IF x>mz T
    HEN y=y+1:x=1                       [5D90]
840 x=MIN(MAX(x,1),mz):y=MIN(MAX(y,1),25
    )                                   [3414]
850 LOCATE x,y:CALL &BB8A:FOR t=1 TO 20:
    NEXT t:CALL &BB8D                   [08B8]
860 IF z$="" THEN 800                   [9626]
870 IF z$=CHR$(127) THEN GOSUB 1010:GOTO
     800                                [3C62]
880 IF z$=CHR$(20) THEN GOSUB 1210:GOSUB
     1440:GOSUB 1080:GOTO 800           [6284]
890 IF ASC(z$)>=240 OR ASC(z$)<32 THEN 8
    00                                  [822C]
900 IF z$=CHR$(16) THEN z$=" "          [BB54]
910 '*************************          [79A2]
920 '** Zeichendarstellung **           [006A]
930 '*************************          [DBA6]
940 PEN pe:PAPER pa                     [DA70]
950 LOCATE x,y:PRINT z$;:POKE 39999+x+mz
    *(y-1),ASC(z$):POKE 40999+x+mz*(y-1)
    ,pe+16*pa                           [8D72]
960 x=x+1:IF x>mz THEN IF y<25 THEN y=y+
    1:x=1                               [8E44]
970 GOTO 800                            [1A62]
980 '*********************              [1CB4]
990 '** Farbroutine PEN **              [2BAA]
1000 '*********************             [B1F4]
1010 WINDOW#0,1,mz,1,1:CLS:INPUT"PEN-Far
     be";pe:pe=pe MOD 16                [1F12]
1020 INPUT"PAPER-Farbe";pa:pa=pa MOD 16 [DBD0]
1030 PEN pe:PAPER pa                    [1FBE]
1040 GOSUB 1120:RETURN                  [C2C6]
1050 '****************************      [1CF6]
1060 '** Darstellungsroutine 1 **       [B28A]
1070 '****************************      [0CFA]
1080 CLS:CALL 42010:PEN pe:PAPER pa:RETU
     RN                                 [EB0E]
1090 '****************************      [34FE]
1100 '** Darstellungsroutine 2 **       [D282]
1110 '****************************      [E9F0]
1120 CLS:z$=""                          [9D3E]
1130 FOR i=1 TO mz:z$=z$+CHR$(14)+CHR$(P
     EEK(40999+i)\16)+CHR$(15)+CHR$(PEEK
     (40999+i))+CHR$(PEEK(39999+i))     [0B24]
1140 NEXT i                             [925C]
1150 WINDOW#0,1,mz,1,25:LOCATE 1,1:PRINT
      z$;                               [CAA4]
1160 PEN pe:PAPER pa                    [34C6]
1170 z$="":RETURN                       [3F44]
1180 '***************************       [0456]
1190 '** Laden und Speichern **         [390C]
1200 '***************************       [3F48]
1210 MODE 1:PEN 1:PAPER 0:CLS           [D2CC]
1220 LOCATE 12,3:PRINT"Funktionswahl:"  [4762]
1230 PRINT:PRINT"{10 SPACE}Laden{8 SPACE
     }(1)"                              [EC48]
1240 PRINT"{10 SPACE}Speichern{4 SPACE}(
     2)"                                [DA38]
1250 PRINT"{10 SPACE}neues Bild{3 SPACE}
     (3)"                               [B930]
1260 PRINT"{10 SPACE}neue Farben{2 SPACE
     }(4)"                              [50B4]
1270 PRINT"{10 SPACE}neue Tasten{2 SPACE
     }(5)"                              [10FA]
1280 PRINT"{10 SPACE}Lader sichern(6)"  [3BEE]
1290 PRINT"{10 SPACE}Verlassen{4 SPACE}(
     7)"                                [BD70]
1300 z$=INKEY$:IF z$="7" THEN 1400      [3EAC]
1310 IF z$="3" THEN GOSUB 470:GOTO 1400 [22D0]
```

Listing 1. »Designer« hilft Bildschirmgrafiken schnell aufzubauen

Farbspeicher. Nach einem Überschreiben müssen wir dann den Bildschirm nur aus dem Zeichenspeicher wieder regenerieren. Zur Darstellung unserer Bilder benutzen wir dabei zwei verschiedene Routinen. Kurze Entscheidungen, wie beispielsweise die Umdefinition der Farben, benötigen nur eine Bildschirmzeile. Dies wollen wir in der obersten Zeile machen. Es muß dann auch immer nur eine Zeile zurücktransferiert werden. Bei umfangreicheren Anweisungen brauchen wir eine Routine, die den gesamten Bildschirm neu aufbaut. Dieses Teilprogramm soll bei jedem Bild mit abgespeichert werden, um die Bilder auch in anderen Programmen benutzen zu können. Damit diese »Datenschaufeleien« schnell genug gehen, müssen wir mit einem Maschinencode-Programm arbeiten.

Auf welche Daten soll diese Maschinencode-Routine zurückgreifen? Zur Ablage des Zeichencodes und der Farbinformation brauchen wie 2000 Byte Speicherplatz. Diesen reservieren wir an der Obergrenze des Benutzerspei-

```
1320 IF z$="4" THEN GOSUB 630:GOTO 1400  [3AD0]
1330 IF z$="5" THEN GOSUB 1490:GOTO 1400
                                         [743E]
1340 IF z$="6" THEN GOSUB 1530:GOTO 1400
                                         [F038]
1350 IF z$<>"1" AND z$<>"2" THEN 1300    [BAF2]
1360 PRINT:PRINT                         [567C]
1370 INPUT"Name des Files";n$            [63F8]
1380 IF z$="1" THEN LOAD n$,40000:mz=PEE
     K(42001):MODE (mz-20)/20:GOSUB 1440
     :RETURN                             [7F56]
1390 POKE 42001,mz:SAVE n$,b,40000,2002  [DCFE]
1400 MODE (mz-20)/20:RETURN              [1138]
1410 '***************************        [714A]
1420 '** Masch-PRG-Daten setzen **       [0194]
1430 '***************************        [674E]
1440 IF mz=20 THEN POKE 42047,&34:POKE 4
     2042,&9E ELSE POKE 42047,&28:POKE 4
     2042,&A0                            [B450]
1450 RETURN                              [8B94]
1460 '**********************             [6F5C]
1470 '** Tastendefinition **             [A326]
1480 '**********************             [9560]
1490 GOSUB 1750:RETURN                   [F1EA]
1500 '******************                 [6156]
1510 '** Lader sichern **                [93D4]
1520 '******************                 [715A]
1530 CLS:INPUT"Name des Laders";n$       [F5FC]
1540 OPENOUT n$+".lad"                   [FB28]
1550 PRINT#9,"10 ' ** Ladeprogramm "+n$+
     " **"                               [E136]
1560 PRINT#9,"20 memory 39999            [ECEA]
1570 z$="25 ":FOR i=0 TO 15:z$=z$+"ink "
     +MID$(STR$(i),2)+","+MID$(STR$(f(i,
     1)),2)+","+MID$(STR$(f(i,2)),2)+":"
     :NEXT i                             [A21C]
1580 PRINT#9,z$                          [9A42]
1590 PRINT#9,"30 DATA 21,40,9c,11,28,a0,
     1a,d5,e5,f5,cd,90,bb                [B00E]
1600 PRINT#9,"40 DATA f1,1f,1f,1f,1f,cd,
     96,bb,e1,e5,7e,cd,5d                [E00A]
1610 PRINT#9,"50 DATA bb,e1,d1,23,13,3e,
     a0,bc,20,e1,3e,28,bd                [9BA8]
1620 PRINT#9,"60 DATA 20,dc,c9,x         [4C42]
1630 PRINT#9,"70 i=42010                 [078A]
1640 PRINT#9,"80 READ a$:IF a$<>"+CHR$(3
     4)+"x"+CHR$(34)+" THEN POKE i,VAL("
     +CHR$(34)+"&"+CHR$(34)+"+a$):i=i+1:
     GOTO 80                             [8A28]
1650 PRINT#9,"50000 '** RUNTIME-MODUL **
                                         [9532]
1660 PRINT#9,"50010 load"+CHR$(34)+n$+CH
     R$(34)+",40000:mz=PEEK(42001):MODE
     (mz-20)/20                          [4A20]
1670 PRINT#9,"50020 IF mz=20 THEN POKE 4
     2047,&34:POKE 42042,&9E ELSE POKE 4
     2047,&28:POKE 42042,&A0             [7AF6]
1680 PRINT#9,"50030 call 42010           [60E2]
1690 PRINT#9,"50040 return               [C7C0]
1700 CLOSEOUT                            [4AAC]
1710 RETURN                              [AE92]
1720 REM *************                   [9420]
1730 REM ** key 464 **                   [1EBC]
1740 REM *************                   [D224]
1750 CLS                                 [229E]
1760 LOCATE 10,3:PRINT"D E S I G N E R{2
      SPACE}4 6 4":PEN 1                 [AB9C]
1770 LOCATE 12,5:PRINT"Tastaturdefinitio
     n"                                  [292E]
1780 PRINT:PRINT:PRINT " Linien / Blockg
     rafik 1{14 SPACE}(1)"               [DF9E]
1790 PRINT" Linien / Blockgrafik 2{14 SP
     ACE}(2)"                            [E648]
1800 PRINT" Blockgrafik 3 + Sonderzeiche
     n{7 SPACE}(3)"                      [5058]
1810 PRINT" griechisches Alphabet+Sonder
     zeichen (4)"                        [FB5C]
1820 PRINT" Grossbuchstaben{21 SPACE}(5)
     "                                   [F55E]
1830 PRINT" Kleinbuchstaben{21 SPACE}(6)
     "                                   [032C]
1840 DATA 67,59,58,50,51,43,42,35,34,27,
     69,60,61                            [8E16]
1850 DATA 53,52,44,45,37,36,71,63,62,55,
     54,46,38                            [2218]
1860 RESTORE 1840                        [FB00]
1870 FOR i=1 TO 26:READ nr(i):NEXT i     [5056]
1880 FOR i=128 TO 159:KEY i,CHR$(i):NEXT
                                         [DF8E]
1890 PRINT" Tastaturbelegung in 'NORMAL'
     -Ebene ";                           [E532]
1900 INPUT n                             [DB10]
1910 IF n>0 AND n<7 THEN 1940            [B14E]
1920 PRINT"Falsche Eingabe !!!"          [3882]
1930 FOR i=1 TO 1500:NEXT i:GOTO 1900    [67A6]
1940 PRINT" Tastaturbelegung{2 SPACE}in
     'SHIFT'-Ebene ";                    [58D4]
1950 INPUT s                             [2024]
1960 IF s<1 OR s>6 THEN 1970 ELSE 1990   [7E86]
1970 PRINT"Falsche Eingabe !!!"          [268C]
1980 FOR i=1 TO 1500:NEXT i:GOTO 1950    [AABA]
1990 PRINT" Tastaturbelegung{2 SPACE}in{
     2 SPACE}'CTRL'-Ebene ";             [A08C]
2000 INPUT c                             [17EA]
2010 IF c<1 OR c>6 THEN 2020 ELSE 2060   [CFFC]
2020 PRINT"Falsche Eingabe !!!"          [0772]
2030 FOR i=1 TO 1500:NEXT i:GOTO 1990    [8BA8]
2040 DATA 129,130,131,132,133,134,135,13
     6,137,138,139,140,141               [D0D8]
2050 DATA 142,143,144,145,146,147,148,14
     9,150,151,152,153,156               [21FE]
2060 DATA 192,193,194,195,196,197,198,19
     9,200,201,202,203,204               [5324]
2070 DATA 205,206,207,208,209,210,211,21
     2,213,214,215,216,217               [94C8]
2080 DATA 218,219,220,221,222,223,226,16
     0,161,162,163,164,165               [10E8]
2090 DATA 166,167,168,169,170,171,172,17
     3,174,175,152,236,237               [B63A]
2100 DATA 176,177,178,179,180,181,182,18
     3,184,185,188,189,190               [4F5A]
2110 DATA 191,203,226,227,228,229,230,23
     1,232,233,234,235,236               [A9E8]
2120 DATA 81,87,69,82,84,89,85,73,79,80,
     65,83,68                            [9160]
2130 DATA 70,71,72,74,75,76,90,88,67,86,
     66,78,77                            [1752]
2140 DATA 113,119,101,114,116,121,117,10
     5,111,112,97,115,100                [F346]
2150 DATA 102,103,104,106,107,108,122,12
     0,99,118,98,110,109                 [7E0E]
2160 FOR i=1 TO 6:FOR j=1 TO 26:READ z(i
     ,j):NEXT j,i                        [5138]
2170 FOR i=1 TO 26:KEY DEF nr(i),1,z(n,i
     ),z(s,i),z(c,i)                     [D348]
2180 NEXT i                              [5C66]
2190 CLS                                 [C39C]
2200 PRINT" Normal : ";:ON n GOSUB 2290,
     2310,2330,2350,2370,2390            [2666]
2210 PRINT" SHIFT{2 SPACE}: ";:ON s GOSU
     B 2290,2310,2330,2350,2370,2390     [06DC]
2220 PRINT" CONTROL: ";:ON c GOSUB 2290,
     2310,2330,2350,2370,2390            [BF84]
2230 PRINT                               [C1E8]
2240 PRINT" O.K. j/n ?"                  [7EAA]
2250 Z$=INKEY$:IF LOWER$(Z$)<>"j" AND LO
     WER$(z$)<>"n" THEN 2250             [C9D0]
2260 IF LOWER$(z$)="n" THEN 1750         [D7C0]
2270 PRINT:PRINT                         [5E7E]
2280 RETURN                              [9F98]
2290 RESTORE 2040:z$="":FOR i=1 TO 26:RE
     AD a:z$=z$+CHR$(a):NEXT:PRINT z$    [D532]
2300 RETURN                              [7E8A]
2310 RESTORE 2060:z$="":FOR i=1 TO 26:RE
     AD a:z$=z$+CHR$(a):NEXT:PRINT z$    [2928]
2320 RETURN                              [928E]
2330 RESTORE 2080:z$="":FOR i=1 TO 26:RE
     AD a:z$=z$+CHR$(a):NEXT:PRINT z$    [4F30]
2340 RETURN                              [FE92]
2350 RESTORE 2100:z$="":FOR i=1 TO 26:RE
     AD a:z$=z$+CHR$(a):NEXT:PRINT z$    [E626]
2360 RETURN                              [C296]
2370 RESTORE 2120:z$="":FOR i=1 TO 26:RE
     AD a:z$=z$+CHR$(a):NEXT:PRINT z$    [9C2E]
2380 RETURN                              [8E9A]
2390 RESTORE 2140:z$="":FOR i=1 TO 26:RE
     AD a:z$=z$+CHR$(a):NEXT:PRINT z$    [CA36]
2400 RETURN                              [8D8C]
```

Listing 1. »Designer« hilft Bildschirmgrafiken schnell aufzubauen (Schluß)

chers, indem wir mit MEMORY die Variable HIMEM verändern. Die Speicheraufteilung finden Sie in Bild 1. Ab Adresse 40000 bis einschließlich 40999 ist der Speicher den Zeichencodes vorbehalten. Jeweils genau 1000 Byte höher liegt die zugehörige Farbinformation.

Die Farbcodierung beinhaltet dabei sowohl die Vorder- als auch die Hintergrundfarbe. Dabei wird ein kleiner Trick angewandt. Zunächst zerlegen wir ein Byte des Farbspeichers in zwei Halbbyte oder auch Nibbles.

Mit vier Bit kann man 16 verschiedene Kombinationen (oder Zahlen von 0 bis 15) darstellen. Das reicht genau für die, maximal im Modus 0 benötigte, Gesamtzahl von 16 Farben aus. Wenn wir nun im unteren Halbbyte die Vordergrund- (PEN) und im oberen die Hintergrundfarbe (PAPER) ablegen, so haben wir in einer 8-Bit-Adresse gleichzeitig zwei Farbregister gespeichert. Dennoch ist für jedes Zeichen Vorder- und Hintergrundfarbe getrennt definiert.

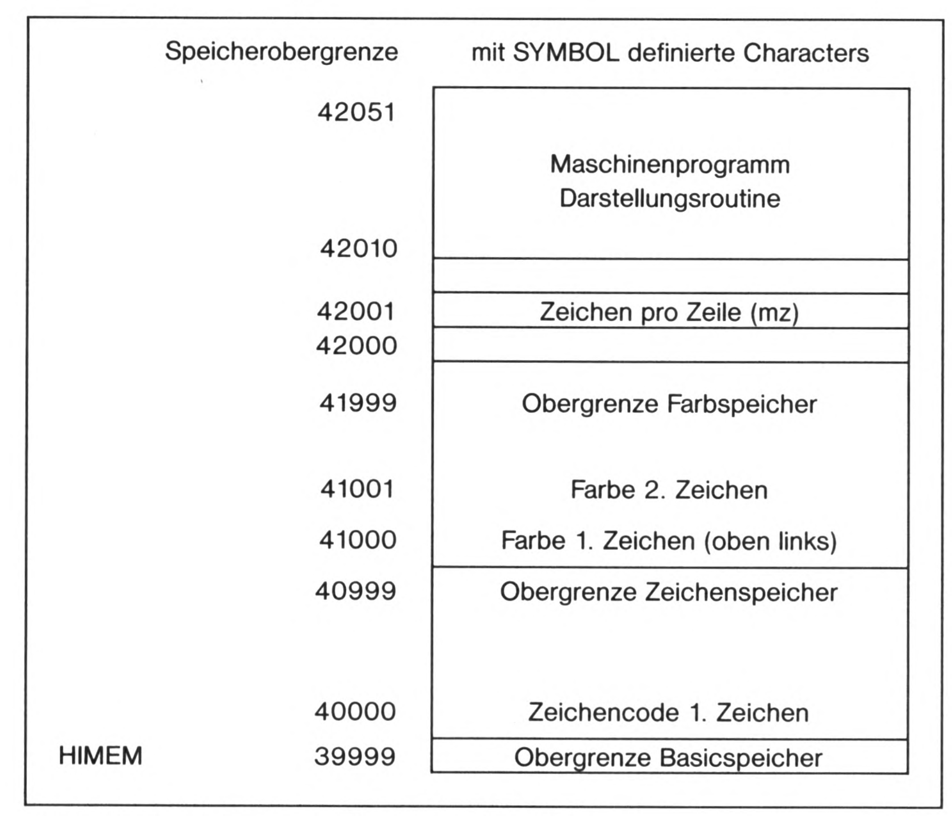


Bild 1. Speicherbelegung des Programms DESIGNER

Eine Besonderheit bietet die Speicherstelle 42001. Sie enthält die aktuelle Anzahl der Zeichen pro Zeile. Wenn wir also im Modus 0 arbeiten, so steht hier eine 20; im Modus 1 eine 40.

```
:HL=Adresse
:DE=Farbe
LD HL,&9C40          21 40 9C
LD DE,&A028          11 28 A0
(BEGINN)
LD A,(DE)            1A
PUSH DE              D5
PUSH HL              E5
PUSH AF              F5
CALL TXT SET PEN     CD 90 BB
POP AF               F1
RRA                  1F
RRA                  1F
RRA                  1F
RRA                  1F
CALL TXT SET PAPER   CD 96 BB
POP HL               E1
PUSH HL              E5
LD A,(HL)            7E
CALL TXT WR CHAR     CD 5D BB
POP HL               E1
POP DE               D1
INC HL               23
INC DE               13
LD A,&A0             3E A0
CP H                 BC
JR NZ (BEGINN)       20 E1
LD A,&28             3E 28
CP L                 BD
JR NZ (BEGINN)       20 DC
RET                  C9
```

Listing 2. Für Maschinencode-Freunde: Das Assemblerlisting

Diese Werte werden von der Darstellungsroutine 1 als Kriterium benutzt, um festzustellen, in welchem Modus das Bild gezeichnet wurde. Ein Grafikbild wird nun gespeichert, indem man den Bereich von Adresse 40000 bis einschließlich 42010 als binäres File sichert.

Die Darstellungsroutine 1 wird hinter die Adresse 42010 gelegt.

Diese Maschinencode-Routine liest jeweils ein Byte aus dem Farb- und aus dem Zeichenspeicher ein, setzt die entsprechenden Vorder- und Hintergrundfarben, und gibt dann das Zeichen auf dem Bildschirm aus.

Für den Umgang und die einfache Benutzung des Blockgrafikentwicklungsprogramms (Designer) ist es nicht notwendig, das Programm in allen Einzelheiten zu verstehen. Wer sich aber für Maschinensprache und Programmierung in Assembler interessiert, findet im Listing 2 den Assembler-Code.

Das Maschinencode-Programm arbeitet mit zwei Zeigern, die in den Registerpaaren HL und DE abgelegt sind. HL enthält dabei die Adresse des darzustellenden Zeichencodes, DE die zugehörige Farbinformation. Nachdem beide Zeiger auf die Anfangswerte (9C40 hex = 40000 beziehungsweise A028 hex = 41000) gesetzt worden sind, beginnt die Ausgabeschleife.

Die Farbinformation wird in A eingelesen, danach werden die Registerpaare DE, HL und AF auf dem STACK gesichert. Es folgt der Aufruf Betriebssystemroutine TXT SET PEN. Diese setzt die Schriftfarbe. Dabei ist zu bedenken, daß TXT SET PEN nur die unteren vier Bit des Akkumulators berücksichtigt, so daß wir uns um die Hintergrundfarbe, die ja in den oberen Bits abgelegt ist, keine Sorgen machen brauchen. Diese Bits werden ignoriert.

Beim Verlassen der Farbdefinitionsroutine sind alle Registerpaare zerstört, das heißt sie enthalten andere Werte. Dies ist dann auch der Grund dafür, daß wir sie vor dem Aufruf der Routine gesichert haben. Zunächst holen wir AF vom Stapel zurück und rotieren den Akkumulator viermal nach rechts.

Damit sind nun die anfangs höherwertigen vier Bit für die Hintergrundfarbe nach unten gerutscht. Sie nehmen jetzt die unteren vier Bit ein und damit dürfen wir TXT SET PAPER aufrufen – eine Routine, die wiederum die übernommenen Register zerstört und den Akkumulator für die Farbdefinition benutzt. Jetzt wird allerdings die Hintergrundfarbe gesetzt. Nach der Rückkehr aus dieser Routine haben wir den Farbdefinitionsteil für das auszugebende Zeichen beendet.

Maschinencode = Geschwindigkeit

Als nächstes wird der Zeichencode bearbeitet. Dazu holen wir HL vom Stapel zurück, geben es aber im nächsten Schritt sofort wieder mit PUSH HL zurück. Dies ist deshalb erforderlich, da auch die dritte nun aufzurufende Betriebssystemroutine TXT WR CHAR die Registerpaare und die Flags verändert. Zuvor laden wir noch den Akkumulator mit dem Inhalt der durch HL adressierten Speicherstelle. Damit steht in A der aktuelle Zeichencode (vergleiche Bild 1). TXT WR CHAR benutzt die zuvor eingestellten PEN- und PAPER-Werte, so daß wir uns um unsere Zeichenausgabe keine weiteren Sorgen zu machen brauchen. Mit der Rückkehr aus dieser Routine ist dann das Zeichen farbrichtig dargestellt.

Als nächstes holen wir uns HL und DE wieder vom Stapel zurück und erhöhen beide um 1. Sowohl der Farbzeiger wie auch die Adresse des Zeichencodes weisen nun auf das nächst höhere Zeichen und wir könnten unsere Schleife für das nächste Zeichen wieder aufrufen. Dies würde sich dann allerdings ewig wiederholen. Wir brauchen also eine Bedingung für das Ende der Bearbeitung.

Dies ist aber einfach zu realisieren. Wir müssen nur überprüfen, ob sämtliche Zeichen ausgegeben worden sind. Dazu kontrollieren wir, ob HL bereits den Wert A028 hex erreicht hat. In letz-

Fortsetzung auf Seite 58



LOGO

Jeder kann programmieren
Computersprache für Eltern und Kinder
DANIEL WATT



"Buch des Jahres 1983" in den USA

LOGO...Ergebnis der Erforschung menschlicher Intelligenz
Entwickelt von Seymour Papert, Pädagoge und Mathematikprofessor.
Erste Computersprache, die bewußt Strategien menschlichen Denkens dient – und in ihrer Logik der Realität gerecht wird.
LOGO ersetzt BASIC, sagen Pädagogen und Mathematiker.
LOGO kommt dem übergreifenden, assoziativen Denken entgegen. BASIC dagegen ist ein Setzkasten von Logik-Buchstaben.

DANIEL WATT... hat im Team von Seymour Papert gearbeitet und ein Buch geschrieben, das voller Bilder seine Erlebnisse mit Kindern am Computer wiedergibt. Ein hochwertiges Textbuch für LOGO-Kurse. Ein Buch für Lehrer, die nach einem bereits von Schulbehörden empfohlenen LOGO-Kursbuch suchen.

Ein Buch für APPLE II, C-64, IBM PC, ATARI bis 520 ST., TI-99 und Schneider CPCs.
384 Seiten, A4, DM 59,–

te-wi
te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

COMPUTER FÜR KINDER

COMPUTER FÜR KINDER Ausgabe APPLE II/IIe
COMPUTER FÜR KINDER Ausgabe Commodore C-64
COMPUTER FÜR KINDER Ausgabe ATARI
Sally G. Larsen

Ein Buch für Kinder und ihre Lehrer – ein kindgemäßes Buch für die erste Begegnung mit Computern, ihren Eigenwilligkeiten, und ihren unerschöpflichen Möglichkeiten. Ein Buch zu unserer Gegenwart und zur Zukunft unserer Kinder. „Computer für Kinder" richtet sich an Kinder im Alter von 8 bis 13 Jahren, für deren Interesse an Computern keines der unzähligen Computer-Bücher geschrieben wurde.

„Computer für Kinder" ist ganz auf Kinder eingestellt und beschäftigt sich unterhaltsam und leicht verständlich mit folgenden Themen:

Wie arbeiten Computer
Wie funktioniert mein Computer
Wie programmiert man mit einfachen Flußdiagrammen
Wie kann ich BASIC leicht verstehen
Programme aufbauen mit Befehlen
Farbige Graphiken entwerfen
Erklärung von Computer-Begriffen

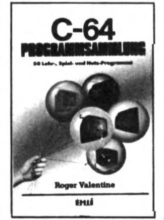
Sally Greenwood Larson war Kindergärtnerin, ehe sie selbst Computern begegnete und zwischen den Welten von Kindern und Computern zu vermitteln begann.

Computer für Kinder, A4 quer, Fadenheftung, über 100 Seiten, je Ausgabe DM 29,80 vorliegend für: VC 20, C 64, Apple II, Atari

Fordern Sie Unterlagen über unser Gesamtprogramm an



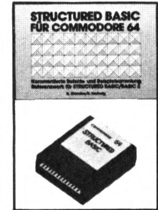
DM 29,80



DM 29,80



DM 59,–



DM 199,–



DM 239,–



DM 49,–



DM 36,–

tm 4416/2

terem Fall wird der komplette Bildschirm wiedergegeben.

Der Befehlssatz des Z80 kennt leider kein Kommando, mit dem der Prozessor in der Lage ist, einen 16-Bit-Vergleich durchzuführen. Wir müssen daher in zwei Stufen überprüfen, ob die Adresse A028 erreicht wurde.

Zunächst vergleichen wir deshalb H mit A0 hex. Wir laden das Register A mit A0 und führen dann einen Vergleich des Akkumulators mit H durch. Ist diese Bedingung schon nicht erfüllt, können wir sofort wieder den Beginn der Schleife aufrufen. Ansonsten laden wir den Akkumulator mit 28 hex und führen den Vergleich mit dem Register L durch. Solange L kleiner als 28 hex ist, geht es wieder zum Startpunkt der Schleife. Im anderen Fall übergibt die Routine wieder an den Basic-Interpreter.

Das »blinkende« Array

Auf der Basis dieses Programms können wir nun die zugehörigen Basic-Befehle für den eigentlichen Designer entwickeln.

Am Anfang unseres Programms finden wir den Initialisierungsteil. Es werden nacheinander die Rahmenfarbe, der Bildschirmdarstellungsmodus und die Farbregister festgelegt. Je INK-Kommando werden dabei zwei Register benötigt, um auch blinkende Farben speichern zu können.

Alle INK-Register sind dabei in dem Array F(15,2) abgelegt. Der erste Index bestimmt das zu definierende Farb-Register, der zweite gibt an, ob es sich um die erste oder zweite zu definierende Farbe handelt. Nachdem das Array mit den Ausgangswerten versehen wurde, werden die INK-Register auf diese Farbwerte gesetzt.

Als nächster Schritt folgt das Einlesen unseres Maschinenprogramms in den Speicherbereich ab Adresse 42010. Dazu wird mit MEMORY die Speicherobergrenze auf 39999 herabgesetzt. Danach kann dann das Maschinenprogramm problemlos und sicher gePOKEt werden.

Der nun folgende Teil lautet salopp »Initialisierungssprünge«. Hier werden nacheinander die eigentlich zu den Sonderfunktionen gehörenden Programmteile zur Tasten-, Farb- und Modusdefinition aufgerufen.

In der Tastatur-Abfrage-Schleife ab Zeile 780 stoßen wir zuerst auf die Cursorbewegungsroutine. In Abhängigkeit von Z$ wird der Cursor entsprechend den Tastencodes von 240 bis 243 bewegt. In Zeile 850 wird dann der Cursor an die aktuelle Bildschirmposition gesetzt. Dies geschieht mit einer Maschinencode-Routine, die an der Adresse BB8A hex ihren Startpunkt hat. Sie setzt den Cursor. Nach einer kurzen Verzögerung durch eine Zeitschleife wird dieser dann mit »CALL&BB8D« wieder gelöscht.

In jedem Fall folgt danach die Rückkehr zu Zeile 800, gegebenenfalls allerdings über einige Umwege. Der einfachste Fall ergibt sich dabei, wenn kein Zeichen eingegeben wurde. In diesem Fall kehrt der Computer sofort wieder an den Schleifenbeginn zurück. Gleiches passiert, wenn ungültige Zeichen, das heißt Zeichen mit einem ASCII-Code, größer als 240 oder kleiner als 32 benutzt werden.

Wird dagegen ein gültiges Zeichen eingegeben, so geht das Programm weiter zum Teil Zeichendarstellung (Zeile 920). Zuerst werden PEN und PAPER auf die Variablen pe beziehungsweise pa gesetzt, die immer den aktuellen PEN- beziehungsweise PAPER-Wert enthalten. Danach wird das gerade eingegebene Zeichen auf dem Bildschirm ausgedruckt.

Gleichzeitig erfolgen zwei relativ kompliziert aussehende POKE-Operationen. Das erste Kommando setzt dabei den Zeichencode. Wir können diesen einfach mit ASC(z$) bestimmen. Mit Hilfe von x und y, die die Spalten und Zeilenzahl auf dem Bildschirm repräsentieren, können wir problemlos die Ausgabeposition im Zeichen- beziehungsweise Farbspeicher errechnen.

Die einzige Schwierigkeit stellt nun bei dieser Art der Speicherung die Bestimmung des Farbwertes dar. Hier sind wir ja mit den zwei Halbbyte konfrontiert. Allerdings ist auch dies kein größeres Problem, da wir einfach den Wert für die Hintergrundfarbe mit 16 multiplizieren und dazu den aktuellen PEN-Wert addieren. Schon liegt der Farbwert in der gewünschten Form vor.

Nach der Überprüfung, ob mit der Zeichenausgabe der rechte Rand des Bildschirms erreicht wurde, kehrt dann das Zeichendarstellungsprogramm in die Hauptabfrageschleife zurück.

Drücken wir statt den Cursorsteuerungstasten oder einer Zeicheneingabetaste auf DEL, so ruft der CPC die Farbroutine ab Zeile 1040 auf. Zeile 900, wo überprüft wird, ob die DEL-Taste gedrückt wurde (CHR$(127)), ist dazu der Initiator.

Die Farbroutine beschränkt zunächst das Ausgabe-WINDOW auf eine Zeile – und zwar die oberste Bildschirmzeile. Dies ist notwendig, um sicherzustellen, daß der Bildschirm nur in diesem Bereich geändert wird. Wir haben zu diesem Zeitpunkt, das heißt beim Aufruf der Farbroutine, immer noch das zu zeichnende Bild auf dem Schirm. Ein unkoordiniertes »Herumirren« gegebenenfalls durch Fehlermeldungen und so weiter verursacht, würde den kompletten Bildschirminhalt zerstören.

Das Verfahren hat darüber hinaus den Vorteil, daß man problemlos und schnell (während der Bildentwicklung) Farbänderungen durchführen kann. Mit INPUT werden die Vorder- und Hintergrundfarbwerte ermittelt. Danach folgt die Darstellungsroutine 2, die die oberste Bildschirmzeile ersetzt. Dazu wird aus den ersten 20 beziehungsweise 40 Farb- und Zeichencodewerten (je nach Modus) ein String zusammengebaut und dieser in der obersten Bildschirmzeile ausgegeben. Er überdeckt damit die gerade für die Farbabfrage ausgegebenen Texte und setzt unseren Bildschirm wieder in den Arbeitszustand zurück.

Nicht ganz so einfach läuft der Prozeß ab, wenn wir auf die Taste »↑« drücken. Am Anfang unseres Programms wurde diese auf CHR$(20) definiert (Zeile 310). Als erster Schritt erfolgt beim Druck auf diese Taste der Aufruf des Hauptfunktionsmenüs ab Zeile 1190.

Die Funktionen 1 und 2 sind einfach zu erklären. Hier findet nur ein Laden beziehungsweise ein Speichern des Adreßbereichs ab Adresse 40000 bis 42010 statt. Mit Funktion 3 (neues Bild) verursachen wir im Prinzip einen Neustart. Der Speicher wird gelöscht und man gelangt zurück in die Modusabfrage ab Zeile 520.

Grafikzeichen selbst definieren

Neue Farben setzen wir mit Funktion 4, die uns zurück in die Definition für die Farb-Register bringt. Eine der wichtigsten Routinen verbirgt sich hinter Funktion 5. Sie werden sich vielleicht schon gefragt haben, wie Sie denn eigentlich die Grafiksymbole, die Sie zwar im Anhang zum Bedienerhandbuch finden, die aber nicht auf der Tastatur verfügbar sind, mit dieser setzen sollen. Nun ganz einfach: Die Tastatur ist nicht mehr wie im Normalfall belegt, sondern verschiedene Grafikzeichen haben jetzt den Platz der Buchstaben eingenommen. Mit Hilfe dieser Funktion greifen Sie auf die neuen Zeichensätze zu. Sie können damit allerdings auch während der Bilderzeugung noch von einem Zeichensatz zum anderen wechseln. Für Ihre praktische Arbeit sind Sie allerdings nicht an diese Zeichen gebunden. Wenn Sie sich mit Hilfe von SYMBOL eigene Zeichensätze definieren, so kann Designer natürlich auch diese benutzen. Allerdings wird der aktuelle Zeichensatz durch das Ladeprogramm nicht mit gesichert. Wenn Sie die mit dem Designer erzeugten Bil-



der in einem anderen Programm benutzen wollen, so müssen Sie dafür sorgen, daß auch diesem die veränderten Symbole zur Verfügung stehen.

Damit haben wir den letzten Teil angesprochen, der im Rahmen dieses Programms behandelt werden soll: den Ladeteil. Er steht unter der Überschrift »Lader sichern« ab Zeile 1510. Wir haben schon gesagt, daß wir zur Darstellung unserer Grafikbildschirme durch ein anderes Basic-Programm ein Ladeprogramm benötigen. Dieses muß drei Funktionen erfüllen:

```
10 ' ** Ladeprogramm test **                   [3BD6]
20 MEMORY 39999                                [1B24]
25 INK 0,0,0:INK 1,24,24:INK 2,6,6:INK 3
   ,2,2:INK 4,21,21:INK 5,15,15:INK 6,7,
   7:INK 7,4,4:INK 8,27,27:INK 9,18,18:I
   NK 10,8,8:INK 11,17,17:INK 12,9,9:INK
    13,11,11:INK 14,26,26:INK 15,24,6:     [3626]
30 DATA 21,40,9c,11,28,a0,1a,d5,e5,f5,cd
   ,90,bb                                  [07C2]
40 DATA f1,1f,1f,1f,1f,cd,96,bb,e1,e5,7e
   ,cd,5d                                  [C8CE]
50 DATA bb,e1,d1,23,13,3e,a0,bc,20,e1,3e
   ,28,bd                                  [916A]
60 DATA 20,dc,c9,x                         [6B02]
70 i=42010                                 [4348]
80 READ a$:IF a$<>"x" THEN POKE i,VAL("&
   "+a$):i=i+1:GOTO 80                     [975C]
50000 '** RUNTIME-MODUL **                 [22EC]
50010 LOAD"test",40000:mz=PEEK(42001):MO
      DE (mz-20)/20                        [49F8]
50020 IF mz=20 THEN POKE 42047,&34:POKE
      42042,&9E ELSE POKE 42047,&28:POKE
       42042,&A0                           [E2AC]
50030 CALL 42010                           [B096]
50040 RETURN                               [E5F2]
```

Listing 3. So muß Ihr Ladeprogramm aussehen

Der richtige Speicher

Zum ersten muß es die Speicherkonfiguration beim Speichern und beim Laden der Bilder herstellen. Dazu genügt ein einfacher Basic-Befehl – nämlich MEMORY 39999. Als zweiten Schritt muß das Ladeprogramm die vom Grafikbildschirm benötigten Farbwerte in die INK-Register schreiben, damit die Darstellung auch in der richtigen Farbe erfolgt. Als dritten Schritt soll unsere Ausgabe-Routine, das heißt die Darstellungsroutine 1, natürlich auch in einem neu zu entwickelnden Basic-Programm vorhanden sein. Denn sonst ist ja keine Ausgabe möglich. Der Aufruf erfolgt dabei jedesmal, wie schon vom Designer gewohnt (Zeile 1080), mit CALL 42010.

Bei der Programmierung dieses Funktionsteils kommt uns eine Besonderheit des Schneiders zugute. Er ist nämlich in der Lage, Programme, die als ASCII-File abgelegt sind, als Programm wieder zu laden und dann auch als Programm auszuführen.

ASCII-Files kennen Sie alle. Wenn Sie schon irgendwann einmal mit OPENOUT eine Datei eröffnet haben (beispielsweise bei jeder Dateiverwaltung), so hatten Sie mit dieser Art der Datenspeicherung zu tun. Dabei werden nämlich die Daten im ASCII-Format Buchstabe für Buchstabe abgelegt, egal ob es sich dabei nun um Strings oder um numerische Variable handelte.

Damit ist auch das Wesentliche bereits gesagt. Wir brauchen nämlich nun nur die Strings, die unsere Programmzeilen enthalten, mit OPENOUT wegzuschreiben, und sie dann mit LOAD als Programm wieder laden. In diesem Fall erkennt der CPC die Strings als Zeilen eines neuen Basic-Programms, setzt diese in seine Basic-Befehle um und ist dann in der Lage, das so entstandene Programm auszuführen.

Wie dies rein technisch vonstatten geht, sehen Sie ab Zeile 1510. Mit PRINT #9 werden nacheinander die einzelnen Basic-Zeilen in die Ausgabedatei geschrieben. Besonders hervorzuheben sind hierbei die Zeilen 25 (1570) und 50010 (1660) des Ladeprogramms.

In Zeile 25 werden die Farb-Register definiert. Sie können hier sehen, wie man ein Programm in einer Ausgabedatei an vom Benutzer eingegebene Werte anpaßt. Die Farbangaben sind ja keineswegs fix, sondern können vom Benutzer laufend verändert werden. Die aktuell zum Zeitpunkt der Abspeicherung vorhandene Farbdefinition soll aber in das Ladeprogramm übernommen werden. Der Lader muß also die Farb-Register auf die aktuellen Werte des Arrays f(15,2) setzen.

STR$ – die Problemlösung

Die Lösung dieses Problems finden Sie in Zeile 1570. Hier wird ein komplexer String für alle 16 Register zusammengebaut. Dazu benutzen wir die STR$-Funktion, welche es uns erlaubt, einen Zahlenwert in einem String zu konvertieren. Diese Strings können wir dann problemlos (auch mit eingefügten Kommas und Doppelpunkten) addieren und erhalten so eine Summe von INK-Befehlen, die nur noch eine vorangestellte Zeilennummer brauchen.

In Zeile 50010 (1660) sehen Sie, wie man ein Laden von einem eingegebenen Benutzer-String abhängig machen kann. Unser Funktionsteil »Lader Sichern« soll nämlich nicht nur für ein ewig feststehendes Ladeprogramm gelten, sondern für beliebige. Wir müssen also zwischen mehreren Namen unterscheiden können, speziell, wenn wir mit der Diskettenstation arbeiten.

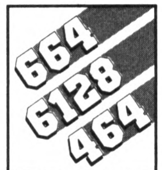
Dies erreichen wir, indem wir den LOAD-Befehl nicht fix ausführen, sondern ihm als Namen eine Benutzereingabe (hier in der Variablen n$ gespeichert) mit auf den Weg geben. Als Kontrollauszug finden Sie nachstehend ein probeweise gesichertes Ladeprogramm (Listing 3). Damit können Sie überprüfen, ob Sie den Laderteil richtig eingegeben haben.

In dem Beispiellisting wurde ein Ladeprogramm in der Ausgangsfarbzusammenstellung gesichert. Rufen Sie sofort nach dem Durchlauf der Anfangsinitialisierung und Druck auf die »↑-Taste« die Funktion 6 auf. Der Rest geschieht automatisch, von ein paar Eingaben Ihrerseits (PLAY und REC) einmal abgesehen.

Arbeiten Sie nun mit Designer und versuchen Sie interessante Titelbilder, Titelgrafiken oder Diagramme zu entwerfen und zu speichern. Ihre Programme werden optisch sicherlich davon profitieren und Bewunderung hervorrufen. Ihr Verständnis für Ihren Schneider wird schnell größer.

(Carsten Straush/hg)

Blitzschnell umgeschaltet



Manchmal reicht ein Zeichensatz allein nicht zur Darstellung aller benötigten Symbole auf dem Bildschirm. Hier ist die eleganteste Lösung des Problems.

Da der Videoprozessor des CPC ständig im Grafikmodus arbeitet, müssen die Zeichen vom Betriebssystem pixelweise auf den Bildschirm verfrachtet werden. Der Vorteil dabei ist, daß man den Zeichensatz an jeder beliebigen Stelle im RAM-Speicher zwischen den Adressen 16384 und 49152 ablegen kann.

Das Betriebssystem merkt sich diesen Bereich in den Speicherstellen B294 hex und B296 hex, um den Zeichensatz wiederfinden zu können. Dazu steht in Adresse B294 hex (beim CPC 664 und CPC 6128 in Adresse B734 hex) der Wert, der bei SYMBOL AFTER angegeben wurde, beziehungsweise direkt nach dem Einschalten 240. In B296 hex (CPC 664 und 6128: B736 hex) steht die Start-Adresse des RAM-Zeichensatzes.

Ändern Sie nun mit »POKE« den Inhalt der Adresse B296 hex so, daß sie auf einen anderen Speicherbereich zeigt, läßt sich in Sekundenbruchteilen zwischen verschiedenen Zeichensätzen wechseln. Bisher mußten die Zeichen dafür mit dem Befehl SYMBOL neu definiert werden.

Der Zeichensatz kann mit »POKE« oder durch Laden eines Binärfiles an seinen Platz gebracht werden. Einfacher ist es jedoch, den Computer am Anfang eines Programms kurzzeitig auf den Speicherbereich für den neuen Zeichensatz umzuschalten, die Zeichen mit »SYMBOL« zu definieren und dann wieder die alten Werte in B296 hex einzutragen. Sie können dann beliebig mit zwei POKE-Befehlen zwischen den Zeichensätzen wechseln.

Das Listing zeigt ein einfaches Beispiel dafür. Wenn das Programm auf einem CPC 664 oder 6128 laufen soll, müssen nur die Adressen wie oben ausgetauscht werden.

(Martin Kotulla/ja)

```
100 ' ********************************     [831C]
110 ' *                              *     [81DA]
120 ' * DEMO: Mehrere Zeichensaetze *     [D38E]
130 ' *        gleichzeitig          *     [4BCE]
140 ' *                              *     [0EE0]
150 ' ********************************     [A626]
160 '                                      [04BC]
170 SYMBOL AFTER 0:MEMORY 32767            [0458]
180 x=PEEK(&B296):y=PEEK(&B297)            [0BBA]
190 POKE &B296,&0:POKE &B297,&80           [DBC2]
200 FOR i=0 TO 255                         [601C]
210 SYMBOL i,RND*255,RND*255,RND*255,RND
    *255,RND*255,RND*255,RND*255,RND*255
                                           [1184]
220 NEXT i                                 [46F8]
230 FOR i=1 TO 10                          [23AE]
240 POKE &B296,x:POKE &B297,y              [88D4]
250 FOR j=0 TO 255:PRINT CHR$(1);CHR$(j)
    ;:NEXT j                               [9466]
260 PRINT:PRINT                            [A218]
270 POKE &B296,&0:POKE &B297,&80           [EAC0]
280 FOR j=0 TO 255:PRINT CHR$(1);CHR$(j)
    ;:NEXT j                               [B06C]
290 NEXT i                                 [A506]
300 POKE &B296,x:POKE &B297,y              [2ACE]
```

Listing. Zwei Zeichensätze im raschen Wechsel

Auf Dauer nur Power



Der CPC 664 und 6128 haben ihrem »kleinen« Bruder CPC 464 einige zusätzliche Grafikbefehle voraus. »Power Graphics« bringt Besitzern des ersten Schneider-Computers eine entsprechende Befehls-Erweiterung.

Mit »Power Graphics« stehen sieben neue RSX-Befehle zur Verfügung. Sie werden von einem senkrechten Balken eingeleitet, der durch gleichzeitigen Druck auf die Tasten »Shift« und »@« erzeugt wird.

Nachdem der Basic-Lader mit »RUN« gestartet wurde und der Computer das Signal »Ready« ausgibt, kann der Basic-Programmspeicher mit »NEW« gelöscht werden. Die Befehls-Erweiterung bleibt bis zum Ausschalten beziehungsweise Reset unverändert erhalten.

IGPEN,f: Dieser Befehl legt mit »f« die Zeichenfarbe für Grafikbefehle fest.

IGPAPER,f: Wie »GPEN«, aber bestimmt die Hintergrundfarbe.

ILINE,x,y,x1,y1,f: Dient zum Zeichnen von Linien und ähnelt somit dem Befehl »DRAW« des Locomotive-Basic 1.0. Im Unterschied dazu braucht man jedoch nicht mit »ORIGIN« den Startpunkt extra zu bestimmen. x und y entsprechen der Anfangs-, x1 und y1 der Endkoordinate und f steht wieder für die Farbe.

ILINER,x,y,x1,y1,f: Ähnliche Wirkung wie »ILINE«. x1 und y1 stehen hier nicht für fixe Koordinaten, sondern für die Distanz zu x und y.

ITRI,x,y,x1,y1,x2,y2,f: x, y, x1, y1, x2 und y2 legen die drei Eckpunkte des Dreiecks fest, das durch diesen Befehl erzeugt wird.

IREC,x,y,x1,y1,f: »REC« zeichnet Rechtecke mit den Koordinaten x und y für die linke untere Ecke, und der horizontalen und vertikalen Ausdehnung, x1, y1.

ISCROLL,r,f: Ist r=0, rollt der Bildschirminhalt um eine Zeile nach unten. Für die Gegenrichtung benötigt r den Wert 1. f steht für die Farbe der nachrückenden Zeile. Die Werte für »f« sollten im Modus 1 entweder 0,15,240 oder 255 sein, da sich ansonsten Streifenmuster bilden.

(Uwe Velker/ja)

```
10  '**********************             [0474]
20  '*                    *             [8DFA]
30  '*     P O W E R      *             [1AD6]
40  '*                    *             [27FE]
50  '*  G R A P H I C S   *             [5EA2]
60  '*                    *             [5202]
70  '*   (C) 1985  by     *             [7B50]
80  '*                    *             [FC06]
90  '*    Uwe Velker      *             [F0FC]
100 '*                    *             [3718]
110 '**********************             [1096]
120 MEMORY &9FFF                        [A4BA]
130 FOR a=&A000 TO &A042:READ x$:POKE a,
    VAL("&"+x$):sum1=sum1+VAL("&"+x$):NE
    XT a                                [D694]
140 IF sum1<>7369 THEN 350              [C288]
150 FOR b=&A043 TO &A0FF:POKE b,0:NEXT b
                                        [8944]
160 FOR c=&A100 TO &A21B:READ x$:POKE c,
    VAL("&"+x$):sum2=sum2+VAL("&"+x$):NE
    XT c                                [DDCA]
170 IF sum2<>42721 THEN 360             [92E0]
180 CALL &A000:END                      [8AFA]
190 DATA 01,09,a0,21,43,a0,c3,d1,bc,20,a
    0,c3,00,a1,c3,0a,a1,c3,14,a1,c3,46,a
    1                                   [D56A]
```

```
200 DATA c3,78,a1,c3,c0,a1,c3,0f,a2,47,5
    0,45,ce,47,50,41,50,45,d2,4c,49,4e,c
    5                                        [4F28]
210 DATA 4c,49,4e,45,d2,54,52,c9,52,45,c
    3,53,43,52,4f,4c,cc,00,00,00,00          [27F0]
220 DATA fe,01,c0,dd,7e,00,cd,de,bb,c9,f
    e,01,c0,dd,7e,00,cd,e4,bb,c9,fe,05,c
    0                                        [1C5E]
230 DATA dd,7e,00,cd,de,bb,cd,c6,bb,d5,e
    5,dd,56,09,dd,5e,08,dd,66,07,dd,6e,0
    6                                        [10E0]
240 DATA cd,c0,bb,dd,56,05,dd,5e,04,dd,6
    6,03,dd,6e,02,cd,f6,bb,e1,d1,cd,c0,b
    b                                        [3400]
250 DATA c9,fe,05,c0,dd,7e,00,cd,de,bb,c
    d,c6,bb,d5,e5,dd,56,09,dd,5e,08,dd,6
    6                                        [3F42]
260 DATA 07,dd,6e,06,cd,c0,bb,dd,56,05,d
    d,5e,04,dd,66,03,dd,6e,02,cd,f9,bb,e
    1                                        [7904]
270 DATA d1,cd,c0,bb,c9,fe,07,c0,cd,c6,b
    b,d5,e5,dd,7e,00,cd,de,bb,dd,56,0d,d
    d                                        [659A]
280 DATA 5e,0c,dd,66,0b,dd,6e,0a,d5,e5,c
    d,c0,bb,dd,56,09,dd,5e,08,dd,66,07,d
    d                                        [E57A]
290 DATA 6e,06,cd,f6,bb,dd,56,05,dd,5e,0
    4,dd,66,03,dd,6e,02,cd,f6,bb,e1,d1,c
    d                                        [1C70]
300 DATA f6,bb,e1,d1,cd,c0,bb,c9,fe,05,c
    0,cd,c6,bb,d5,e5,dd,7e,00,cd,de,bb,d
    d                                        [53E8]
310 DATA 56,09,dd,5e,08,dd,66,07,dd,6e,0
    6,d5,e5,cd,c0,bb,dd,56,05,dd,5e,04,d
    d                                        [895E]
320 DATA 66,03,dd,6e,02,e5,d5,11,00,00,c
    d,f9,bb,d1,21,00,00,cd,f9,bb,e1,cd,c
    7                                        [F146]
330 DATA bd,11,00,00,cd,f9,bb,e1,d1,cd,f
    6,bb,e1,d1,cd,c0,bb,c9,fe,02,c0,dd,4
    6                                        [C6E6]
340 DATA 02,dd,7e,00,cd,4d,bc,c9             [8316]
350 CLS:PRINT CHR$(7);"Fehler in den ers
    ten 66 DATA's":END                       [AA9E]
360 CLS:PRINT CHR$(7);"Fehler in den DAT
    A's 67 bis 283":END                      [8E76]
```

Listing. Grafik-»Power« bringt dieser Basic-Lader

Wandernde Balken



Ein kleines Maschinencode-Programm läßt in Sekundenbruchteilen den Grafikbildschirm invertieren. Damit sind erstaunliche Effekte zu erzielen.

Wenn Sie den Basic-Lader aus Listing 1 eintippen und starten, meldet der Computer auf dem Bildschirm die Adresse des CALL-Aufrufs für die Maschinencode-Routine. Das Ladeprogramm legt den Maschinencode immer direkt unter HIMEM ab und setzt dann HIMEM um 20 Byte herunter. Solange Sie nicht mit »MEMORY«, »SYMBOL AFTER« oder durch das Öffnen einer Kassetten- oder Disketten-Datei etwas an der oberen Speichergrenze ändern, können Sie das Programm auch mit »CALL HIMEM+1« aufrufen.

Das Maschinencode-Programm Listing 2 invertiert in kürzester Zeit alle Bildschirm-Punkte. So wird aus dem Byte 255 jetzt Null und umgekehrt. Wenn Sie im Modus 2 arbeiten, wird die Zeichenfarbe gegen die Hintergrundfarbe ausgetauscht. In den beiden anderen Bildschirm-Modi ändern sich die Farben nach etwas komplizierteren Regeln, auf deren Erläuterung wir hier aber verzichten wollen.

Im Lader ist auch eine Demonstration des Maschinencode-Programms integriert. Durch eine optische Täuschung glaubt der Betrachter, auf dem Bildschirm würden ständig Balken abwärts wandern – wie gesagt: nur eine Täuschung!

(Martin Kotulla/ja)

```
100 ' ***************************           [9F78]
110 ' *                         *           [0B9A]
120 ' * Bildschirm invertieren  *           [19B4]
130 ' *                         *           [FB9E]
140 ' ***************************           [4780]
150 '                                       [E1BA]
160 MEMORY HIMEM-20:FOR i=HIMEM+1 TO HIM
    EM+18:READ a:POKE i,a:NEXT i            [0442]
170 MODE 1:PEN 1:LOCATE 12,10:PRINT "Auf
    ruf: CALL &";HEX$(HIMEM+1)              [C59E]
180 LOCATE 19,15:SPEED INK 30,30:INK 3,1
    ,24:PEN 3:PRINT "DEMO"                  [D0D4]
190 FOR i=1 TO 5000:NEXT i                  [7D84]
200 DATA &21,&00,&C0,&11,&FF,&FF,&7E,&2F
    ,&77                                    [B468]
210 DATA &23,&E5,&B7,&ED,&52,&E1,&20,&F5
    ,&C9                                    [CF74]
220 ' DEMO-PROGRAMM:                        [8ED8]
230 MODE 1                                  [C156]
240 INK 0,1:INK 1,24:INK 2,20:INK 3,6:IN
    K 4,1                                   [1EBC]
250 FOR i=1 TO 25                           [1ABE]
260 PEN i MOD 4                             [CAF0]
270 PRINT STRING$(40,143);                  [7D2A]
280 NEXT i                                  [4F04]
290 FOR i=1 TO 1000:CALL HIMEM+1:NEXT i     [A802]
```

Listing 1. Der Bildschirm »kippt«

```
                  10  ;****************************
                  20  ;* Bildschirm-Invertierung  *
                  30  ;****************************
                  40
A000              50            org   #a000       ;Programmstart
A000   2100C0     60  start:    ld    hl,#c000    ;Bildschirmanfang
A003   11FFFF     70            ld    de,#ffff    ;Bildschirmende
A006   7E         80  inv_lp:   ld    a,(hl)      ;Byte lesen
A007   2F         90            cpl               ;invertieren
A008   77         100           ld    (hl),a      ;wieder speichern
A009   23         110           inc   hl          ;naechstes Byte
A00A   E5         120 cphlde:   push  hl          ;vergleiche >hl< mit >de<
A00B   B7         130           or    a
A00C   ED52       140           sbc   hl,de
A00E   E1         150           pop   hl
A00F   20F5       160           jr    nz,inv_lp   ;bis fertig
A011   C9         170           ret               ;Ruecksprung ins Basic
A012              180           end
```

Listing 2. Quell-Code zum Invertieren

Seidenweiche Bildschirmverschiebung II



Windows haben die Schneider CPCs ja ohnehin. Was bisher fehlte, war eine effektvolle Nutzung. »Omega-Basic« sorgt für Abhilfe mit butterweichem Scrolling.

In Happy- Computer, Ausgabe 9/85 beschrieb ein Artikel pixelweises Scrolling. Aber damals ging es darum, den gesamten Bildschirminhalt zu bewegen. Diesmal jedoch sollen beliebige Teilbereiche in Marsch gesetzt werden. Und das sogar in alle Richtungen. »OMEGA-Basic« heißt das Zauberwort zum Öffnen der Schatztruhe.

Listing 1 enthält den Basic-Lader für ein Maschinencode-Programm zur Erzeugung zehn zusätzlicher RSX-Befehle. Bevor Sie den ersten Probelauf durchführen, sollten Sie das Programm speichern. Nach korrektem Lauf legt es die Befehls-Erweiterung selbsttätig als Binär-Datei »OMEGA.BIN« ab. Für den späteren Gebrauch benötigen Sie den Basic-Lader nicht mehr. Es genügt dann, die RSX-Befehle mit »MEMORY &8FFF : LOAD "OMEGA.BIN" : CALL &9000« zu laden und aktivieren. Die neuen Befehle beginnen, wie bei RSX (Resident System Extension) üblich, mit einem senkrechten Balken, der durch gleichzeitigen Druck auf die Tasten »Shift« und »@ « entsteht. Darauf folgen die eigentlichen Befehlsworte, die sich in vier Kategorien fassen lassen.

»Omega-Basic« – Eine Befehlserweiterung für Feinschmecker

Die erste Gruppe, »IROLLR«, »IROLLL«, »IROLLU« und »IROLLD«, läßt den Inhalt des gewünschten Bildschirm-Bereiches rollen; besser gesagt er »rotiert«. Was an einem Ende aus dem Bereich wandert, wird an der gegenüberliegenden Seite wieder angefügt. Der letzte Buchstabe des Befehls steht dabei für die Richtung, in die gerollt werden soll. Genauso verhält es sich mit der zweiten Gruppe: »ISCROLLR«, »ISCROLLL«, »ISCROLLU« und »SCROLLD«. Hierbei gehen die nach außen wandernden Informationen jedoch verloren. Nur je einen Befehl umfassen die beiden letzten Kategorien. »ICLW« löscht den Inhalt des Bildschirmbereiches, »IINVERSE« invertiert (Hintergrund- wird Vordergrundfarbe und umgekehrt) ihn.

Natürlich haben Sie sich schon gefragt, woher der Computer wohl wissen soll, welcher Bereich denn nun gemeint ist. Selbstverständlich können Sie ihm genau das mitteilen, indem Sie an jeden der aufgeführten Befehle beim Aufruf vier Werte übergeben. Listing 2 zeigt sehr eindrucksvoll die Syntax und Wirkungsweise. Die einzelnen Werte stehen, in der Reihenfolge der Aufzählung, für X-Ordinate, Y-Ordinate, Breite und Höhe des Bild-Ausschnittes. Die Wirkung sehen Sie am besten, wenn Sie selbst ein bißchen mit dem Beispielprogramm experimentieren.

Die Routinen sind auf den Bildschirm-Modus 2 ausgelegt. Da die Farbcodierung in den anderen beiden Modi abweicht, ergeben sich dort bei horizontalen Bewegungen farbliche Veränderungen, die aber durchaus ihren Reiz haben können. Wichtig für eigene »Omega«-Programme ist nur, daß vor dem Bildaufbau durch einen MODE-Befehl die Bildschirm-Basis festgelegt wird.

(Heiko Falkenhof und Matthias Wagner/ja)

```
10 'Basic Loader Omega                          [C314]
20 'Heiko Falkenhof & Matthias Wagner, T
   empelberg 20, 2153 Neu Wulmstorf             [9326]
30  adr=&9000                                   [A50C]
40 MEMORY adr-1                                 [4B24]
50 FOR n=0 TO 58                                [396E]
60 checksumme=0                                 [0430]
70 FOR a=1 TO 15                                [C14C]
80 READ wert                                    [BC0C]
90 POKE adr+n*15+a-1,wert                       [8320]
100 checksumme=checksumme+wert                  [384A]
110 NEXT a                                      [1BE4]
120 READ kontrollsumme                          [8516]
130 IF checksumme<>kontrollsumme THEN PR
    INT"Datafehler in Zeile ";(n+1)*10+1
    000:END                                     [90B8]
140 NEXT n                                      [3404]
150 SAVE "Omega",B,adr,&36B                     [5200]
160 END                                         [EF1C]
1010 DATA 0,205,21,147,1,13,144,33,105,1
     44,195,209,188,45,144, 1594                [2712]
1020 DATA 195,94,146,195,43,146,195,79,1
     47,195,93,147,195,52,145, 2067             [DAB4]
1030 DATA 195,122,145,195,219,145,195,6,
     147,195,249,145,195,158,146, 2457          [79D0]
1040 DATA 83,67,82,79,76,76,204,83,67,82
     ,79,76,76,210,83, 1423                     [5A92]
1050 DATA 67,82,79,76,76,213,83,67,82,79
     ,76,76,196,82,79, 1413                     [8CB4]
1060 DATA 76,76,204,82,79,76,76,210,82,7
     9,76,76,213,82,79, 1566                    [1604]
1070 DATA 76,76,196,73,78,86,69,82,83,19
     7,67,76,215,0,201, 1575                    [7506]
1080 DATA 0,0,0,0,13,191,32,66,65,83,73,
     67,32,49,46, 717                           [A0DE]
1090 DATA 48,32,32,32,32,32,32,164,32,49
     ,57,56,53,32,72, 755                       [A27A]
1100 DATA 101,105,107,111,32,70,97,108,1
     07,101,110,104,111,102,32, 1398            [DE52]
1110 DATA 38,32,77,97,116,116,104,105,97
     ,115,32,87,97,103,110, 1326                [402E]
1120 DATA 101,114,13,10,202,232,144,243,
     254,4,194,18,145,22,0, 1696                [2AEC]
1130 DATA 221,126,6,50,227,144,79,221,12
     6,2,50,229,144,6,175, 1806                 [56BA]
1140 DATA 129,218,18,145,128,218,18,145,
     221,126,4,50,228,144,79, 1871              [5608]
1150 DATA 221,126,0,50,230,144,6,230,129
     ,218,18,145,128,218,18, 1881               [6278]
1160 DATA 145,201,0,0,80,1,201,62,0,50,2
     27,144,50,228,144, 1533                    [204C]
1170 DATA 62,80,50,229,144,62,25,50,230,
     144,201,33,176,191,58, 1735                [D926]
1180 DATA 228,144,198,1,71,17,80,0,25,16
     ,253,58,227,144,95, 1557                   [FA22]
1190 DATA 22,0,25,201,33,36,145,6,16,126
     ,205,90,187,35,16, 1143                    [877C]
1200 DATA 249,30,255,22,255,251,201,13,7
     9,117,116,32,111,102,32, 1865              [85C6]
1210 DATA 83,99,114,101,101,110,13,10,20
     5,169,144,122,254,255,200, 1980            [F480]
1220 DATA 205,251,144,43,14,8,58,230,144
     ,87,229,58,229,144,71, 1915                [0342]
1230 DATA 229,213,22,0,88,25,126,23,119,
     43,16,250,245,22,0, 1421                   [C6CE]
1240 DATA 58,229,144,95,25,241.48,2,203,
     198,209,225,213,17,80, 1987                [4558]
1250 DATA 0,25,209,21,32,216,225,213,17,
     0,8,25,209,13,32, 1245                     [C5FE]
1260 DATA 201,251,201,205,169,144,122,25
     4,255,200,205,251,144,14,8, 2624           [A4EC]
1270 DATA 58,230,144,87,229,58,229,144,7
     1,229,126,31,119,35,16, 1806               [BEBA]
1280 DATA 250,225,48,2,203,254;213,17,80
     ,0,25,209,21,32,231, 1810                  [172E]
1290 DATA 225,213,17,0,8,25,209,13,32,21
     6,251,201,205,169,144, 1928                [E918]
1300 DATA 122,254,255,200,205,251,144,17
     ,0,160,58,230,144,71,14, 2125              [25B2]
1310 DATA 8,229,205,234,145,213,17,0,8,2
     5,209,13,32,244,225, 1807                  [DB4C]
1320 DATA 213,17,80,0,25,209,16,232,251,
     201,205,176,145,123,254, 2147              [2BC6]
1330 DATA 255,200,33,0,160,205,234,145,2
     51,201,197,229,58,229,144, 2541            [7CA4]
1340 DATA 6,0,79,237,176,225,84,93,193,2
     01,205,169,144,122,254, 2188               [64B4]
```

Listing 1. Der Basiclader erzeugt die Befehlserweiterung

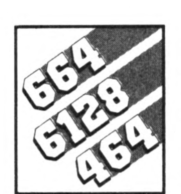
```
1350 DATA 255,200,205,251,144,14,8,58,23
     0,144,87,229,58,229,144, 2256          [1506]
1360 DATA 71,229,126,47,119,35,16,250,22
     5,213,17,80,0,25,209, 1662             [DFC4]
1370 DATA 21,32,235,225,213,17,0,8,25,20
     9,13,32,220,251,201, 1702              [B61C]
1380 DATA 205,169,144,122,254,255,251,20
     0,205,251,144,14,8,58,230, 2510        [A99E]
1390 DATA 144,87,229,58,229,144,71,229,1
     26,31,119,35,16,250,225, 1993          [AA2A]
1400 DATA 213,17,80,0,25,209,21,32,235,2
     25,213,17,0,8,25, 1320                 [4EF2]
1410 DATA 209,13,32,220,251,201,205,169,
     144,122,254,255,200,205,251, 2731      [D042]
1420 DATA 144,43,14,8,58,230,144,87,229,
     58,229,144,71,229,213, 1901            [D444]
1430 DATA 22,0,88,25,126,23,119,43,16,25
     0,22,0,58,229,144, 1165                [AE86]
1440 DATA 95,25,209,225,213,17,80,0,25,2
     09,21,32,222,225,213, 1811             [ADA0]
1450 DATA 17,0,8,25,209,13,32,207,251,20
     1,205,169,144,87,254, 1822             [18BC]
1460 DATA 255,200,205,251,144,14,8,58,23
     0,144,87,229,58,229,144, 2256          [040A]
1470 DATA 71,229,54,0,35,16,251,225,213,
     17,80,0,25,209,21, 1446                [C384]
1480 DATA 32,236,225,213,17,0,8,25,209,1
     3,32,221,201,205,169, 1806             [99A6]
1490 DATA 144,122,254,255,200,205,251,14
     4,58,230,144,71,17,80,0, 2175          [BCDA]
1500 DATA 25,16,253,17,176,55,25,17,0,16
     0,71,14,8,229,205, 1271                [618C]
1510 DATA 234,145,213,17,0,8,237,82,209,
     13,32,243,225,213,17, 1888             [6FC2]
1520 DATA 80,0,237,82,209,16,230,251,201
     ,205,206,146,122,254,255, 2494         [C03E]
1530 DATA 200,33,0,160,205,234,145,251,2
     01,62,2,205,14,188,6, 1906             [7190]
1540 DATA 60,33,109,144,126,205,90,187,3
     5,16,249,6,5,33,255, 1553              [7F72]
1550 DATA 255,43,124,181,32,251,16,246,1
     ,224,1,62,1,50,230, 1717               [13D4]
1560 DATA 144,197,205,101,146,193,11,120
     ,177,32,246,62,2,205,14, 1855          [7BE6]
1570 DATA 188,62,201,50,0,144,201,205,17
     6,145,122,254,255,200,33, 2236         [4A30]
1580 DATA 80,160,205,234,145,201,205,206
     ,146,123,254,255,200,33,80, 2527       [2DFE]
1590 DATA 160,205,234,145,201,0,0,0,0,0,
     0,0,0,0,0, 945                         [69A6]
```

Listing 1. Der Basiclader erzeugt die Befehlserweiterung (Schluß)

```
10 ' DEMO Programm   Omega                   [5236]
20 ' Heiko Falkenhof & Matthias Wagner       [E818]
30 MEMORY &8FFF:LOAD"Omega":CALL &9000       [9C30]
40 MODE 2                                    [13F6]
50 FOR n=1 TO 9:PRINT STRING$(18,CHR$(19
   1)):NEXT                                  [6364]
60 FOR n=1 TO 3:FOR m=1 TO 8                 [266C]
70 |ROLLR,0,0,21,12                          [CFC8]
80 |ROLLD,0,0,21,12                          [9DAE]
90 NEXT m,n                                  [9DDE]
100 LOCATE 6,3:PRINT CHR$(191);" Basic 1
    .0"                                      [0F44]
110 FOR N=1 TO 728:|ROLLR,2,2,18,1:NEXT      [25C2]
120 FOR n=1 TO 240                           [DC1E]
130 |ROLLU,3,3,9,9                           [788C]
140 |ROLLR,3,3,18,4                          [8BDE]
150 |ROLLL,3,8,18,4                          [65DE]
160 |ROLLD,12,3,9,9                          [B1D0]
170 NEXT                                     [69EE]
180 FOR N=1 TO 221                           [EFE8]
190 |ROLLU,3,3,9,9                           [A098]
200 |ROLLD,12,3,9,9                          [91C6]
210 NEXT                                     [6CE4]
220 |CLW,9,6,6,3                             [E63C]
230 FOR n=1 TO 3                             [355C]
240 FOR m=1 TO 48                            [0FCE]
250 |ROLLR,3,3,6,9                           [9186]
260 |ROLLR,3,3,6,9                           [9188]
270 |ROLLL,15,3,6,9                          [A8E4]
280 |ROLLL,15,3,6,9                          [A1E6]
290 |ROLLD,3,3,18,3                          [8ECC]
300 |ROLLU,3,9,18,3                          [7AEA]
310 NEXT m                                   [2E00]
320 |INVERSE,3,3,18,9                        [DA0A]
330 |CLW,9,6,6,3                             [D540]
340 NEXT                                     [77EC]
350 |INVERSE,3,3,18,9                        [1510]
360 FOR n=1 TO 32                            [3AC8]
370 |SCROLLL,3,3,6,9                         [D7AC]
380 |SCROLLL,3,3,6,9                         [1FAE]
390 |SCROLLR,15,3,6,9                        [FC22]
400 |SCROLLR,15,3,6,9                        [FB12]
410 |SCROLLU,3,3,18,3                        [5F0E]
420 |SCROLLD,3,9,18,3                        [04FA]
430 NEXT                                     [6EEC]
440 LOCATE 40,8:PRINT"Press any key to e
    nd"                                      [E636]
450 |CLW,3,3,18,9                            [DCA6]
460 IF INKEY$<>"" THEN END                   [4522]
470 GOTO 350                                 [DC58]
```

Listing 2. Experimentieren Sie mit »Omega-Basic«

Mit Tricks an ROM-Routinen



Es gibt auch in Basic eine Vielzahl nützlicher ROM-Routinen. Wichtige Routinen wollen aber Parameter, und das ist von Basic aus normalerweise nicht möglich.

Mit einem Trick geht es doch. Bedingung ist, daß der einzige zu übergebende Wert im Akkumulator (A-Register) steht und der Wert kleiner als 33 ist.

Dazu muß man wissen, daß der Basic-Interpreter erlaubt, bei CALL-Befehlen bis zu 32 Parameter zu übergeben, zum Beispiel:

»CALL &A000,3,4,@a$,-32«

Der Routine wird dann im IX-Register die Basisadresse der gespeicherten Parameter übergeben und im Akku die Anzahl der Parameter, die dem CALL-Aufruf folgen.

Und das ist auch schon der ganze Trick: Durch eine Anzahl von Dummy- oder Leer-Argumenten wird der Akku mit deren Zahl geladen und der ROM-Routine übermittelt.

Nützliche Anwendungen lassen sich leicht dafür finden. Beispielsweise fehlt dem CPC 464 der, bei den Nachfolgemodellen vorhandene, GRAPHICS-PEN- und GRAPHICS-PAPER-Befehl, der sich jedoch recht einfach simulieren läßt:

»GRAPHICS PEN 0« = »CALL &BBDE«
»GRAPHICS PEN 1« = »CALL &BBDE,0«
»GRAPHICS PEN 2« = »CALL &BBDE,0,0«
»GRAPHICS PEN 3« = »CALL &BBDE,0,0,0«
»GRAPHICS PEN 4« = »CALL &BBDE,0,0,0,0«

GRAPHICS PEN setzt die Vordergrundfarbe für die Grafikausgabe analog zum PEN-Befehl, der nur auf die Textausgabe wirkt. GRAPHICS PAPER steuert die Farbe des Grafikfensters und wird zum Beispiel nach »CLG« sichtbar:

»GRAPHICS PAPER 0« = »CALL &BBE4«
»GRAPHICS PAPER 1« = »CALL &BBE4,0«
»GRAPHICS PAPER 2« = »CALL &BBE4,0,0«
»GRAPHICS PAPER 3« = »CALL &BBE4,0,0,0«
»GRAPHICS PAPER 4« = »CALL &BBE4,0,0,0,0«

Diese CALL-Aufrufe funktionieren auch auf dem CPC 664 und CPC 6128, so daß Anwender, die austauschbare Programme schreiben wollen, statt der GRAPHICS-Befehle besser auf diese Routinen zurückgreifen. Kompatibilität ist also durchaus erreichbar.

(Martin Kotulla/ja)

Flimmerkiste



Wer wünscht sich nicht, Grafiken »ruck-zuck« auf dem Bildschirm zu haben. Dabei kann das Gegenteil davon manchmal erheblich reizvoller sein.

Das Programm »Blenden« zeigt Effekte, die durch Benutzung der Register des Video-Controllers möglich sind.

Der HD 6845 hat insgesamt 17 Register. Die Register bis einschließlich 11 sind lediglich beschreibbar, während die Register 12 bis 15 gelesen und beschrieben und die Register 16 bis 17 nur gelesen werden können. Man erreicht sie, indem man über Schnittstellen-Adresse BC00 hex mit einem OUT-Befehl die Nummer des Registers angibt und es danach mit »IN« oder »OUT« über Adresse BD00 hex liest oder schreibt. Im Klartext:

```
10 input »Registernummer«;r:input »Wert«;w
20 out &bc00,r:out &bd00,w
30 goto 10
```

ermöglicht ein Beschreiben des angegebenen Registers,

```
10 input »Registernummer«;r
20 out &bc00,r:in &bd00,w:? »Wert =«;w
30 goto 10
```

das Lesen. Sie können nun sehr eindrucksvoll Bilder ineinander übergehen lassen und so effektvolle Spiele programmieren. (Dieter Braun/ja)

```
5 GOTO 20000                                   [D640]
10 REM    I Runterschieben **************
   *****                                       [D346]
20 FOR n=25 TO 0 STEP-1                        [60D0]
30 OUT &BC00,6:OUT &BD00,n                     [6500]
40 OUT &BC00,7:OUT &BD00,n+5:CALL &BD19        [F0DC]
50 NEXT:CALL &BD19                             [FCA0]
55 OUT &BC00,6:OUT &BD00,0                     [7092]
60 OUT &BC00,7:OUT &BD00,0                     [838C]
70 RETURN                                      [77CE]
100 REM    II Hochholen ******************
    *****                                      [AFF6]
140 FOR n=0 TO 25                              [2BC2]
150 OUT &BC00,6:OUT &BD00,n                    [6766]
160 OUT &BC00,7:OUT &BD00,n+5:CALL &BD19
                                               [8D42]
180 NEXT:RETURN                                [EB24]
200 REM    III  Ueberblenden *************
    *****                                      [128C]
210 FOR n=25 TO 0 STEP-1                       [C432]
220 OUT &BC00,6:OUT &BD00,n:CALL &BD19         [D27A]
230 NEXT                                       [78E8]
250 RETURN                                     [AD2E]
300 REM    IV   Aufblenden von oben *****
    *****                                      [70CA]
310 OUT &BC00,7:OUT &BD00,30                   [F94E]
340 FOR n=0 TO 25                              [33C6]
350 OUT &BC00,6:OUT &BD00,n:CALL &BD19         [D182]
360 NEXT                                       [6BF0]
370 RETURN                                     [A634]
400 REM    V   Auf Mitte ueberblenden ***
    *****                                      [1FD0]
405 z=30                                       [46A6]
410 FOR n=25 TO 0 STEP-2                       [6838]
420 OUT &BC00,7:OUT &BD00,z:z=z-1              [B712]
430 OUT &BC00,6:OUT &BD00,n:CALL &BD19         [8C80]
440 NEXT                                       [66EE]
450 OUT &BC00,6:OUT &BD00,0:                   [4A64]
460 RETURN                                     [A934]
500 REM    VI  Von Mitte aufblenden *****
    *****                                      [6D4C]
505 z=18:REM Hier Bild aufbauen                [2044]
510 FOR n=0 TO 25 STEP 2                       [D620]
520 OUT &BC00,7:OUT &BD00,z:z=z+1              [F310]
530 OUT &BC00,6:OUT &BD00,n:CALL &BD19         [D382]
540 NEXT                                       [75F0]
550 OUT &BC00,6:OUT &BD00,25                   [3360]
560 RETURN                                     [AA36]
600 REM    VII Seitwaerts Blenden *******
    *****                                      [B33A]
610 FOR n=40 TO 0 STEP-1                       [D834]
620 OUT &BC00,1:OUT &BD00,n                    [C060]
625 FOR m=0 TO 5:NEXT                          [015E]
630 NEXT                                       [84F0]
640 RETURN                                     [1B34]
700 REM    VIII Seitwaerts Blenden auf **
    *****                                      [8FE2]
710 FOR n=0 TO 40                              [4CC2]
720 OUT &BC00,1:OUT &BD00,n                    [B062]
725 FOR m=0 TO 5:NEXT                          [F760]
730 NEXT                                       [D3F2]
740 RETURN                                     [AC36]
800 REM    IX  Seitwaerts auf Mitte Blend
    en **                                      [B2A6]
810 z=46                                       [7AB4]
820 FOR n=40 TO 1 STEP-2                       [9F3E]
830 OUT &BC00,1:OUT &BD00,n                    [DF66]
840 OUT &BC00,2:OUT &BD00,z:CALL &BD19         [909A]
850 z=z-1                                      [9498]
860 NEXT                                       [76FA]
870 OUT &BC00,1:OUT &BD00,0                    [86F2]
880 RETURN                                     [C140]
900 REM    X  Seitwaerts von Mitte aufble
    nden                                       [5B14]
910 z=26                                       [73B2]
920 FOR n=0 TO 40 STEP 2                       [7524]
930 OUT &BC00,1:OUT &BD00,n                    [C168]
940 OUT &BC00,2:OUT &BD00,z:CALL &BD19         [7F9C]
950 z=z+1                                      [0E96]
960 NEXT                                       [1DFC]
970 RETURN                                     [2040]
1000 REM    XI In Mitte Blenden *********
     ****                                      [86B4]
1010 GOSUB 400:GOSUB 800                       [5010]
1020 RETURN                                    [8586]
1100 REM    XI Von Mitte aufblenden *****
     ****                                      [0216]
1110 GOSUB 900:GOSUB 500                       [9016]
1120 RETURN                                    [8688]
1200 REM    XII  Erdbeben / Explosion ein
      ***                                      [F524]
1210 OUT &BC00,8:OUT &BD00,1                   [A74C]
1220 RETURN                                    [AB8A]
1250 REM    XII  Erdbeben / Explosion aus
      ***                                      [8E48]
1260 OUT &BC00,8:OUT &BD00,2                   [AC58]
1270 RETURN                                    [8994]
1300 REM    XIII Eckblenden *************
     ****                                      [06A0]
1310 FOR n=40 TO 0 STEP-1                      [0B90]
1320 OUT &BC00,1:OUT &BD00,n                   [E0BC]
1330 OUT &BC00,6:OUT &BD00,INT(n*0.625):
     CALL &BD19                                [99A2]
1340 NEXT                                      [F84E]
1350 RETURN                                    [AA92]
1400 REM    XIV  Eckaufblenden **********
     ****                                      [8DE6]
1410 FOR n=0 TO 40                             [6B1E]
1420 OUT &BC00,1:OUT &BD00,n                   [60BE]
1430 OUT &BC00,6:OUT &BD00,INT(n*0.625):
     CALL &BD19                                [63A4]
1440 NEXT                                      [E750]
1450 RETURN                                    [8B94]
20000 MODE 1:LOCATE 16,2:PRINT CHR$(24);
      "EFFEKT";CHR$(24)                        [3F4A]
20010 PRINT:PRINT:BORDER 0                     [5E5E]
20020 PRINT"Durch direkten Zugriff auf d
      ie Register des Video-Controllers
      koennen tolle{5 SPACE}Effekte erzi
      elt werden.":PRINT                       [A4F2]
20030 PRINT"Um ein Bild aufzubauen kann
      z.B. die An-leitung davor mit eine
      r schicken Blende mit der Borderfa
      rbe ueberschrieben wer- den, das B
      ild im Dunkeln aufgebaut und{2 SPA
      CE}danach mit einer Blende wieder
      gezeigt{2 SPACE}werden."                 [AB14]
20040 PRINT:PRINT"Oder man laesst das Bi
      ld bei einer{6 SPACE}Explosion erz
      ittern."                                 [10D8]
20060 LOCATE 13,25:PRINT"Taste druecken.
      "                                        [1324]
20070 CALL &BB18                               [ABD0]
20080 MODE 1:LOCATE 4,12:PRINT"Blende Nu
      mmer 1"                                  [6A3E]
20090 FOR n=0 TO 3000:NEXT                     [3A38]
21000 d=1:en=8                                 [EDCE]
21010 ON d GOSUB 10,200,400,600,800,1000
      ,1300                                    [936E]
21020 MODE 1:LOCATE 4,13:PRINT"Blende Nu
      mmer ";d+1                               [0ACA]
21030 ON d GOSUB 100,300,500,700,900,110
      0,1400                                   [CFDE]
21035 FOR n=0 TO 3000:NEXT                     [2138]
21040 d=d+1:IF d=en THEN 21100                 [3970]
21050 GOTO 21010                               [4ECA]
21100 MODE 1:LOCATE 6,13:PRINT"Bildzitte
      rn"                                      [15F0]
21110 FOR n=0 TO 2000:NEXT                     [472A]
21120 GOSUB 1200                               [3BF2]
21130 FOR n=0 TO 5000:NEXT                     [1D34]
21140 GOSUB 1250                               [6B00]
21150 MODE 1:LOCATE 6,13:PRINT"Das war's
       dann ....":CALL &BB18                   [B760]
21160 MODE 2:END                               [CC44]
```

Listing. Szenenwechsel »Wie im Kino«.



Von CALL, RSX und anderen Spezialitäten



Zwar ist das Basic des Schneiders reichhaltig, aber es gibt drei Wege, den Sprachumfang des Interpreters zu erweitern.

Das Basic des Schneider zählt – speziell wenn man es mit anderen Dialekten im Heimcomputer-Bereich vergleicht – zu einer der stärksten Versionen. Dennoch stellt man bereits nach relativ kurzer Eingewöhnungszeit einige Mängel fest. So fehlt beispielsweise ein Befehl, der das Zeichnen von Kreisen erleichtert. Es gibt zwar eine Reihe vorgefertigter Algorithmen für solche und ähnliche Problemstellungen und für die meisten Standardprobleme existieren auch bereits fertige Programme in Z80-Maschinencode. Das Problem besteht nun aber darin, diese an den Schneider anzupassen, und zwar möglichst komfortabel. Eine Erweiterung sollte dabei möglichst wie ein normaler Basic-Befehl angesprochen werden. Dazu müssen wir aber unser Maschinencode-Programm irgendwie mit dem Interpreter verbinden. Um den Wortschatz des Interpreters zu erweitern, gibt es beim Schneider mehrere Wege.

Der CPC kennt zwei verschiedene Grundvarianten, um ohne Eingriffe in die Hardware Maschinencode-Routinen von Basic aus aufrufen zu können: den Aufruf eines Systemprogrammes mit CALL und die Einbindung eines Maschinenprogrammes in das Betriebssystem mit Hilfe einer sogenannten RSX. Dieses Kürzel steht für **R**esident **S**ystem **E**xtension, was übersetzt als bleibende Systemerweiterung interpretiert werden kann. Daneben existiert beim 464 noch eine indirekte Art der Einbindung, die sich besonders für größere Spracherweiterungen eignet. Dieser Weg führt über die sogenannten Error-Vektoren.

Wir wollen uns die verschiedenen Varianten anhand einer kleinen Basic-Spracherweiterung anschauen.

Einführend befassen wir uns mit dem Aufruf eines Maschinencode-Programms mit CALL und RSX. Der CALL-Befehl hat das Format »CALL <Ansprungadresse>,Wert 1,Wert 2, ..., Wert n«, wobei die Wertangaben optional sind. Insgesamt können maximal 32 16-Bit-Werte übergeben werden. Sie können aber auch entfallen. Trifft der Basic-Interpreter unseres Schneiders auf dieses Kommando, so ruft er die Routine auf, die ab der angegebenen Adresse vorliegt. Die übergebenen Daten befinden sich beim Aufruf in einem Zwischenspeicher, dem sogenannten Stack. Die Register der CPU enthalten dabei die zur Datenabfrage und Interpretation notwendigen Informationen.

Befehlserweiterung mit Komfort: RSX

Zum Vergleich schauen Sie sich den Aufruf einer RSX an. Die RSX stellt eine Erweiterung des Betriebssystems dar. Ihr Aufbau ist um einiges komplizierter als der Aufruf eines einfachen CALL-Befehls. Allerdings erhält man als Lohn für seine Anstrengungen auch einen

erheblich verbesserten Bedienkomfort.

Im Prinzip handelt es sich bei der RSX um ein Maschinencode-Programm, genauer eine Serie von Maschinencode-Routinen, über die ein Programmkopf in einem bestimmten Format so gestülpt wurde, daß die Programme durch den Interpreter aufgerufen und ausgeführt werden können. Dementsprechend erfolgt der Aufruf der Erweiterung innerhalb einer RSX dann auch ähnlich wie die Ansprache eines Basic-Kommandos. Das Format lautet dabei »<Name der RSX-Routine>,Wert 1,Wert 2, ... ,Wert n« und ähnelt damit stark dem CALL-Aufruf. Auch die Datenübergabe erfolgt auf dieselbe Art und Weise. Bevor wir uns näher mit den Unterschieden zwischen RSX und CALL-Kommando beschäftigen, müssen wir uns zunächst einmal der eigentlichen Datenübergabe, das heißt dem Zustand der CPU-Register und des Stack beim Aufruf dieser Programme, zuwenden.

Die Datenübergabe erfolgt indiziert

Die Register A und B sowie das Indexregister IX stehen in direktem Zusammenhang mit der Anzahl der übermittelten Daten. A beinhaltet ihre Anzahl. Es können maximal 32 Werte an eine Maschinencode-Routine oder RSX übergeben werden. B gibt die Differenz zwischen dem tatsächlichen und dem Maximalwert 32 oder 20 hex an.

Die einzelnen übermittelten Daten sind dabei auf einem Stapelspeicher, dem Stack, abgelegt. IX gibt die Adresse des untersten Elements dieses Speichers an. Jeder Wert, den wir den Befehlen mitgeben, wird in eine 16-Bit-Binärzahl umgewandelt, diese wiederum byteweise (zuerst die 8 niederwertigen Bits und dann die 8 höherwertigen) auf dem Stapel abgelegt. Der erste übergebene Wert liegt zuoberst. Mit zunehmenden Werten dehnt sich der Stapel dann immer weiter nach unten aus, womit auch IX kleinere Werte enthält. Machen wir uns das Zusammenwirken der drei Register nun einmal anhand eines Beispiels klar.

Dazu betrachten wir ein ganz normales CALL-Kommando: »CALL 40000,250,1000«.

Wie sehen nun unsere Registerinhalte aus? Für A und B ist diese Frage einfach zu beantworten. Es wurden 2 Parameter übergeben. A enthält folglich eine 2 und B eine 30 (32 minus 2). Der Stapel sieht dabei wie folgt aus:

```
00
FA
03
E8
```

IX deutet nun auf die letzte Adresse dieses Speichers, also die mit dem Wert E8. Die benutzten hexadezimalen Werte erhalten Sie, indem Sie sich mit der Funktion HEX$ das hexadezimale Äquivalent der nach dem CALL-Kommando stehenden Zahlen ausgeben lassen.

»PRINT HEX$(250,4)« liefert als Ergebnis 00FA. Führen wir denselben Befehl mit dem Wert 1000 durch, so erhalten wir als Ergebnis 03E8. Diese Werte schreiben wir nun einfach von oben nach unten in unseren Stapel und erhalten damit den oben beschriebenen Stapelaufbau.

Wie können wir aber nun die im Stack stehenden Daten in die Register unseres Prozessors (der CPU) laden, damit wir damit in unseren Maschinencode-Programmen arbeiten können.

Das ist relativ einfach. Der Z80-Prozessor kennt nämlich Befehle, die das indizierte Laden eines Registers der CPU aus der durch das Indexregister IX adressierten Speicherstelle erlaubt. Wenn wir also unsere 1000 in den Registern H und L speichern wollen, so genügen zwei einfache Kommandos.

```
LD H,(IX+1)
LD L,(IX+0)
```

Damit befinden sich die beiden Bytes dieser Zahl als 16-Bit-Wert in dem Registerpaar HL gespeichert. Dasselbe Verfahren können wir natürlich nun auch verwenden, um die 250 in das Registerpaar DE laden. Nach

```
LD D,(IX+3)
LD E,(IX+2)
```

ist unsere Datenübertragung perfekt, und die auszuführende Maschinencode-Routine, die auf diese Daten zurückgreifen soll, kann aufgerufen werden. An deren Ende muß dann der Maschinensprache-Befehl RET (C9 hex) stehen, der den Computer anweist, ins Basic zurückzukehren.

Wir haben jetzt alle Informationen, die wir brauchen, um an ein Maschinencode-Programm Daten mit CALL zu übergeben und diese dann ausführen zu lassen. Wir wollen deshalb beide Varianten, den CALL-Befehl wie auch die RSX, anhand eines konkreten Beispiels durchexerzieren.

Als Beispiel schreiben wir uns eine Maschinencode-Routine, die den Grafikspeicher umschaltet. Die Umschaltung der Bildschirmbasis geht am einfachsten mit der Firmware-Routine SCR SET BASE. Diese erfordert in Register A das signifikante Byte der Bildschirmadresse. Man kann zwar auch zwei Routinen schreiben, wobei die erste auf den unteren Grafikspeicher umschaltet, die andere auf den oberen; wenn man aber einen Entscheidungsparameter mit übergibt, reicht auch nur eine Routine aus. Legen wir unser Beispielprogramm ab Speicherstelle 40000 ab, so lautet der Aufruf: »CALL 40000, <Inhalt Register A>«, wobei der einzige Parameter durch das signifikante, höherwertige Byte der Bildschirm-Basisadresse zu ersetzen ist. Wollen wir auf den unteren Grafikspeicher umschalten, so müßte hier der Wert 40hex eingegeben werden, ansonsten C0 hex. Die nachstehenden niederwertigen 8 Bit finden durch SCR SET BASE ja keine Berücksichtigung.

Ein Maschinencode-Programm, das auf die Dateneingabe reagiert, ist dann problemlos zu schreiben. Es benötigt nur drei Befehle:

```
LD A,(IX+0)          DD 7E 00
CALL SCR SET BASE    CD 08 BC
RET                  C9
```

Die hexadezimalen Codes DD 7E stehen für das Laden des Akkumulators aus der durch IX adressierten Speicherstelle. Der dritte Wert gibt die Differenz zur gesuchten Speicherstelle an (in unserem Fall keine = 0). Wenn Sie diese sieben Werte nach »MEMORY 39999« mit »POKE 40000,&DD:POKE 40001,&7E:POKE 40002,&00:POKE 40003,&CD:POKE 40004,&08:POKE 40005,&BC:POKE 40006,&C9« in den Speicher ab der Adresse 40000 laden, so können Sie mit »CALL 40000,&40« auf den unteren Speicher schalten und mit »CALL 40000,&C0« wieder zurückkehren. (Der Aufruf der beiden Bildspeicherbereiche erfolgt nur korrekt, wenn zwischenzeitlich der Bildschirm nicht geSCROLLt wurde.)

Zweiter Grafikspeicher

Dies ist die einfachste Variante des CALL-Befehls. Sie dürfen nach demselben Prinzip natürlich auch mehrere Variablen übermitteln. Speziell, wenn Sie mit verschiedenen CALL-Befehlen in ein und demselben Programm arbeiten, werden Sie möglicherweise die Ansprungadressen verwechseln oder auch die zu übermittelnden Daten. Hier gibt es wieder eine relativ einfache psychologische Hilfe. Sie definieren nämlich Variable entsprechend der Namen der Maschinencode-Routinen und laden diese mit der Aufrufadresse. Dann können Sie ein Programm auch mit Namen aufrufen. Wenn Sie beispielsweise die Variablen

```
Umschaltung=40000
Oberspeicher=&C0
Unterspeicher=&40
```

festlegen, so können Sie mit »CALL Umschaltung,Unterspeicher« auf den unteren Grafikspeicher schalten, beziehungsweise mit »CALL Umschaltung, Oberspeicher« wieder den normalen, oberen Grafikspeicher ab Adresse

C000 hex benutzen. Funktional hat sich bei dieser Art des Aufrufes nichts geändert. Der einzige Unterschied besteht darin, daß durch diesen Trick der CALL-Befehl etwas »menschenfreundlicher« gestaltet wurde.

Wir können unseren Aufruf jedoch noch komfortabler gestalten, indem wir die Umschaltung zwischen den Bildschirmen als Basic-Kommando mit Hilfe der RSX ausführen.

Beginnen wir mit einigen grundsätzlichen Bemerkungen zur Ablage der RSX. Eine resistente Systemerweiterung ist immer im RAM, genauer in den zentralen 32 KByte RAM abgelegt. Dies ist der erste Unterschied zum CALL-Befehl, der den gesamten RAM-Speicher unseres Computers adressieren kann.

Während wir also mit CALL jede Adresse zwischen 0 und 65535 dez als Startwert eines Maschinencode-Programmes benutzen dürfen, sind wir bei der Verwendung des RSX-Befehls auf die Adressen von 4000 bis BFFF hex beschränkt. Eine RSX legt man daher meist an der Speicherobergrenze ab.

Woraus besteht nun eine RSX? Um diese Frage zu beantworten, gehen wir am besten von der Art des Aufrufes aus. Wie schon gesagt, schafft eine Systemerweiterung mit RSX neue Befehle (Basic-Erweiterungs-Befehle). Der Aufruf einer RSX-Routine geschieht daher auch mit einem Basic-ähnlichen Befehl.

Unser Beispielprogramm könnten wir mit den Kürzeln OGRAPH (für das Umschalten auf die obere Grafik) und UGRAPH (für die Bewegung in der Umkehrrichtung) bezeichnen. Wenn wir dann in einem Basic-Programm diese Befehle mit dem immer noch voranzustellenden Erweiterungsstrich (»|«) eingeben, so soll der Schneider automatisch auf den gewünschten Speicherbereich umschalten.

Das setzt natürlich voraus, daß der Computer die zugehörige Ausführungsroutine kennt und weiß, daß diese irgend etwas mit den beiden Kommandos zu tun hat. Wir müssen das Betriebssystem also zuerst von der Existenz dieser Routine in Kenntnis setzen: Die RSX muß initialisiert werden.

Ist eine RSX-Funktion einmal in das Betriebssystem eingebaut worden, so kann man sie immer wieder im Verlauf des Programms ohne weitere Informationen an das Betriebssystem aufrufen. Alles funktioniert genauso, wie bei einem normalen Basic-Befehl.

Die Initialisierungsprozedur erfolgt dabei mit Hilfe einiger Routinen aus dem Betriebssystemkern, den Kernal-Routinen. Wir wollen uns die Einbindung der RSX in das Betriebssystem einmal anhand unserer beiden Befehle anschauen.

Eine RSX besteht aus zwei großen Teilen
- einem Sprungzeigerteil und dem Maschinencode-Programm.

Der Sprungzeigerteil enthält die Aufrufadressen aller Maschinencode-Routinen, die zur RSX gehören. Eine Tabelle, die angibt, bei welchen Namen welche Routine aufgerufen werden soll, findet sich auch in diesem Teil.

Im zweiten Teil stehen die eigentlichen Programme. Wir können die auszuführenden Maschinencode-Routinen in beliebiger Reihenfolge (gegebenenfalls auch ineinander verschachtelt) an jeder Stelle im RAM positionieren.

Etwas anders sieht es dagegen mit unserer Sprungtabelle aus. Diese darf,

A000	LD HL,0000	21 00 00
A003	LD (A010),HL	22 10 A0
A006	LD BC, <RSX-Anfang= A020>	01 20 A0
A009	LD HL, <Kernal-Speicher=A035>	21 35 A0
A00C	JP KL LOG EXT	C3 D1 BC
A010	Zwischenspeicher OFFSET LOW	
A011	Zwischenspeicher OFFSET HIGH	
A020	Anfang Namenstabelle	28 A0
A022	JP UGRAPH	C3 40 A0
A025	JP OGRAPH	C3 60 A0
A028	U	55
A029	G	47
A02A	R	52
A02B	A	41
A02C	P	50
A02D	H + 80 HEX	C8
A02E	O	4F
A02F	G	47
A030	R	52
A031	A	41
A032	P	50
A033	H + 80 HEX	C8
A034	Trennungsnull	00
A035	Kernal-Speicher 3 Bytes frei	
A040	'PRG:UGRAPH	
A040	CALL SCR GET LOCATION	CD 0B BC
A043	EX DE,HL	EB
A044	LD A,&40	3E 40
A046	CALL SCR SET BASE	CD 08 BC
A049	LD HL,(A010)	2A 10 A0
A04C	CALL SCR SET OFFSET	CD 05 BC
A04F	LD (A010),DE	ED5310A0
A053	RET	C9
A060	'PRG:OGRAPH	
A060	CALL SCR GET LOCATION	CD 0B BC
A063	EX DE,HL	EB
A064	LD A,&C0	3E C0
A066	CALL SCR SET BASE	CD 08 BC
A069	LD HL,(A010)	2A 10 A0
A06C	CALL SCR SET OFFSET	CD 05 BC
A06F	LD (A010),DE	ED5310A0
A073	RET	C9

Tabelle 1. RSX-Sprungblock

wie schon gesagt, nur in den zentralen 32 KByte des Speichers liegen, also nicht unterhalb eines ROMS. Normalerweise wird man sie, wie schon von der CALL-Routine gewohnt, an der Speicherobergrenze – also zum Beispiel im Bereich von A000 hex oder auch 40000 dez – ablegen.

Die Sprungzeigertabelle besteht aus zwei Teilen: dem eigentlichen Sprungzeigerblock und den Namen der neuen Befehle. Für den Aufbau dieser beiden Blöcke gelten eine Reihe von Konventionen, die unbedingt eingehalten werden müssen; sonst kann der Betriebssystemkern, das Kernal, die Maschinencode-Routinen nicht in den Basic-Interpreter einbinden und später auch nicht ausführen. Sie finden den Aufbau eines RSX-Sprungblocks in Tabelle 1 am Beispiel unserer beiden Befehle.

Den Anfang dieses Maschinensprache-Listings bildet die Initialisierungsroutine für die RSX. Wir kommen auf diesen Programmteil gleich noch zurück. Ab Adresse A020 hex finden Sie dann die eigentliche RSX-Sprungtabelle.

In den ersten 2 Byte unseres Sprungblocks steht eine Adresse, der Zeiger weist auf den Anfang der Namenstabelle. Danach folgen als Sprung-Kommandos (JP) die Aufrufadressen der verschiedenen Routinen. Diese sind im 3-Byte-Format hintereinander abgelegt. Der Aufruf erfolgt jeweils durch einen unbedingten direkten Sprung (JP).

Ab der Adresse, auf die der Zeiger der Namenstabelle deutet, liegen dann die einzelnen Befehlsnamen. Ein solcher darf aus einem oder mehreren Buchstaben und/oder Punkten bestehen. Leerschritte innerhalb des Namens sind nicht zugelassen, da der Computer den Leerschritt als Trennmarkierung zwischen einzelnen Basic-Kommandos im Interpreter benutzt.

Würden wir also einen Befehl mit Leerschritt definieren, so würde der Interpreter ihn als zwei neue Basic-Kommandos erkennen. Wenn wir davon ausgehen (was der wahrscheinlichste Fall ist), daß die Einzelwörter keinen Sinn ergeben, also als einzelne Kommandos nicht definiert sind, ergibt sich ein »Syntax Error«.

Die ASCII-Codes der Namen sind nun hintereinander – beginnend mit dem ersten Namen – abzulegen. Dabei muß allerdings noch ein Punkt beachtet werden. Da Befehlsnamen und Variablennamen beim Schneider keine vorgegebene Länge haben, muß dem Computer irgendwie mitgeteilt werden, daß der Name beziehungsweise die Namensdefinition beendet ist.

Dies geschieht, indem man zum letzten Zeichen des Schlüsselwortes 128

oder 80 hex addiert. Dadurch wird das höchste Bit (Bit 7) gesetzt, was der CPC als Endmarkierung wertet. Aus diesem Grund ist auch klar, daß Grafikzeichen (mit Werten über 128) in Befehlswörtern nichts zu suchen haben. Der CPC würde diese in ein, um 128 im ASCII-Code niedriger liegendes Zeichen, übersetzen und das Befehlswort als abgeschlossen betrachten. Da er mit den nachfolgenden Zeichen nichts mehr anfangen könnte, ergäbe sich wiederum eine Fehlermeldung. Den Abschluß der Namenstabelle bildet ein Byte mit dem Wert 00 hex.

Nach dem Ende der Tabelle müssen noch 4 Byte für den Betriebssystemkern reserviert werden. Hier stellt der Computer die Verbindung zum Basic-Interpreter her.

Wenn Sie diese Konventionen beachten, ist die Initialisierung der RSX kein Problem mehr. Der Anfang unserer RSX-Befehlstabelle muß in das Registerpaar BC geladen werden. Die Adresse des ersten für das Kernal reservierten Bytes gehört in HL. Wenn Sie jetzt noch die Betriebssystemroutine KL LOG EXT aufrufen, ist Ihre Arbeit beendet. Die neuen Routinen sind als Erweiterungskommandos integriert. Den Rest leistet der CPC selber.

Die dazu notwendigen Befehle finden Sie am Anfang unseres Assemblerlistings. Die ersten beiden Befehle gehören dabei nicht zur Initialisierungsroutine. Sie löschen vielmehr die Speicherstellen A010 und A011. In diesen Adressen werden unsere beiden Maschinencode-Programme den letzten Offset (Abstand zwischen Startadresse (Basis) des Bildspeichers und Adresse des ersten Zeichens) zwischenspeichern. Um eine definierte Anfangsposition zu erhalten, sind diese beiden Adressen am Anfang auf 0 zu setzen.

Damit sind wir nun auch bei der Erklärung der eigentlichen Routinen angelangt. Komplizierter aufgebaut als die einfachen Versionen, die mit CALL arbeiteten, weisen sie dafür aber einen bedeutend höheren Komfort auf.

Ab Adresse A040 hex finden Sie die Befehle für das Kommando UGRAPH. Zunächst wird eine Betriebssystemroutine aufgerufen – SCR GET LOCATION. Diese stellt eine kombinierte Umkehrroutine zu SCR SET BASE und SCR SET OFFSET dar. Sie liest die aktuellen Werte für Basis und Offset und gibt diese in Register A (signifikantes Byte der Basis) beziehungsweise in das Registerpaar HL (Offset) zurück.

Als nächsten Schritt sichern wir durch Vertauschen der Registerinhalte HL in DE und laden dann den Akkumulator mit 40 hex. Es folgt der Aufruf von SCR SET BASE und damit die Umschaltung auf den unteren Grafikspeicher. Damit ist vorerst unsere Arbeit beendet und wir können ins Basic zurückkehren.

Auch mit Patches geht's

Ein guter Verschiebebefehl muß es jedoch ermöglichen, mit verschiedenen Werten für den Offset zu operieren. Wenn beispielsweise auf dem Hauptbildschirm ein Scroll ausgeführt wurde, so würde ohne Berücksichtigung des Offset auch der Inhalt unseres Nebenspeichers neu interpretiert werden. Es fände auch hier eine Verschiebung statt. Dies darf jedoch nicht passieren, wenn man permanent zwischen zwei Schirmen hin- und herschalten will, um dort die verschiedenen Ausgaben zu positionieren (wobei ja nicht jedesmal der Schirm gelöscht werden darf).

Daher setzen UGRAPH und OGRAPH auch gleichzeitig noch den Offset neu – und zwar auf den alten Wert, der in dem nun aktuell ausgewählten Grafikspeicher vor der letzten Umschaltung Geltung hatte.

Dazu wird der letzte Offset, der in A010 und A011 hex zwischengespeichert war, in HL geladen. Als nächstes setzt SCR SET OFFSET den Offset-Zeiger auf den in HL gespeicherten Wert. Damit ist nun auch der Offset-Pointer korrigiert.

Wem dieser Offset ein Buch mit sieben Siegeln ist, der sollte sich an das Scrollen des Bildschirms beim Schneider erinnern. Normalerweise stimmt die Startadresse des Bildspeichers mit der Adresse des oberen linken Punkts überein. Wird allerdings der Bildschirm um eine Zeile nach oben geschoben (gescrollt), so wird einfach zu der Startadresse des Bildspeichers ein Wert (Offset) addiert, so daß das Zeichen der zweiten Zeile angesprochen wird. Startadresse des Bildspeichers und Adresse des ersten Zeichens stimmen also nicht mehr überein, sondern sind durch den Offset getrennt. Jedesmal, wenn sich das Bild verschiebt, ändert sich dieser Offset. Zurück zu unserem Programm:

Nach der Umschaltung auf den alten Offset-Wert muß natürlich der neue, gerade überschriebene Wert gesichert werden, denn bei einer Rückschaltung brauchen wir diesen ja wieder. Am Anfang unserer Routine hatten wir ihn in DE gesichert. Wir laden nun unseren Zwischenspeicher an der Adresse A010 hex aus diesem Registerpaar und können jetzt mit RET wieder ins Basic zurückkehren.

Die Interpretation des Befehls UGRAPH ist damit abgeschlossen. Um das Kommando OGRAPH ausführen zu können, müssen wir in unserer Befehlskette nur eine einzige Angabe ändern, nämlich den Wert der Bildschirmbasis. A muß dazu mit C0 hex geladen werden. Das restliche Programm bleibt gleich.

Unsere Befehlserweiterung ist damit komplett. Sie muß nur noch im Speicher Platz finden. Dazu dient ein kleines Ladeprogramm (Listing). Nachdem Sie das Programm eingetippt haben, sollten Sie es vor dem ersten RUN-Versuch unbedingt speichern. Bereits ein kleiner Fehler in einem DATA-Statement kann dazu führen, daß sich der Computer »aufhängt«. Die RSX-Befehle werden nach »RUN« mit »CALL &A000« (CALL<Startadresse Initialisierungsroutine) eingebunden.

Sie sollten nun ruhig ein wenig mit der RSX-Erweiterung spielen und sich überlegen, wie Sie eigene Maschinencode-Programme auf das RSX-Format bringen können. Als Ausgangsbasis Ihrer Überlegungen kann Ihnen Tabelle 1 dienen.

Zu Beginn dieses Artikels haben wir erwähnt, daß es noch eine dritte Art der Einbindung von Maschinencode-Routinen in den Basic-Raum gibt – die Befehlsinterpretation mittels der Error-Vektoren. Die ersten beiden Varianten sind durch das Betriebssystem vorgesehen und auch vorbereitet und binden eine erlaubte Erweiterung in das System ein. Die Error-Vektoren ähneln eher dem Basic-Kommando ON ERROR GOSUB und sind eher für Pro-

```
10 '************************              [55DC]
20 '** graphikswitch rsx **               [060C]
30 '**      by C.S.      **               [C07E]
40 '************************              [93E2]
50 MEMORY &4000-1                         [C78C]
60 DATA 21,00,00,22,10,a0,01,20,a0,21,35
   ,a0,c3,d1,bc,x                         [FFCA]
70 DATA 28,a0,c3,40,a0,c3,60,a0,55,47,52
   ,41,50,c8,4f,47,52,41,50,c8,00,00,00,
   00,x                                   [48BC]
80 DATA cd,0b,bc,eb,3e,40,cd,08,bc,2a,10
   ,a0,cd,05,bc,ed,53,10,a0,c9,x          [F268]
90 DATA cd,0b,bc,eb,3e,c0,cd,08,bc,2a,10
   ,a0,cd,05,bc,ed,53,10,a0,c9,x          [8BC8]
100 FOR k=0 TO 3                          [2A4C]
110 FOR i=k*&20+&A000 TO k*&20+&A020      [3D84]
120 READ a$:IF a$="x" THEN 140 ELSE POKE
    i,VAL("&"+a$)                         [B6F4]
130 NEXT i                                [23F8]
140 NEXT k                                [2DFE]
150 FOR i=&A000 TO &A100:PRINT HEX$(i,2)
    ,HEX$(PEEK(i),2):NEXT                 [CBF2]
```

Listing. Der Basic-Lader für unsere RSX-Befehle

Kennen Sie auch die Schneider-Ausgaben 1 und 2 von »Happy-Computer«?

Nein?

Dann sollten Sie diese beiden Sonderhefte unbedingt kennenlernen!

Denn mit Schneider 1 schaffen Sie sich ein ausführliches und umfassendes Grundlagenwissen.



Das erste große Schneider-Sonderheft

Lernen Sie die fast unbegrenzten Einsatzmöglichkeiten der tollen CPC-Schneider-Computer kennen. Vergrößern Sie den Spaß an Ihrem Hobby und reißen Sie Ihren Computer noch mehr aus. Über 120 Seiten Hilfe. Alle Schneider-Computer – CPC 6128, 664, 464 – im Vergleich. Grafik: »Geheimcodes« zur Bildschirmgestaltung. Listing: Malen wie auf einer Leinwand. Sound: Ihr Schneider spielt Bach/Musik und Sound selbst programmiert. Anwendungen: Echtzeitverarbeitung auf dem Schneider / Assembler-Disassembler für den CPC464. Tips&Tricks: Deutsche Tastatur für Ihren Schneider.

Schneider 2 hält viele weiterführende Informationen, Anwendungen, Tips und Tricks parat.



Das zweite große Schneider-Sonderheft

Wichtige Tips und Tricks für Einsteiger und Fortgeschrittene: U.a. selbsttätiges Kopieren der gesamten Diskette auf Kassette / Neuer RSX-Befehl »Circle« / Zeitersparnis: Speichern im Block. Listings: Disk-Doktor / Krimi-Adventure »Famit AG« / »Supermon CPC-1002« / Alle Listings mit Prüfsumme. Grundlagen: So programmiert man 3-D-Grafik / Die interessantesten Firmware-Routinen. Preiswert selbstgebaut: RS232-Schnittstelle – mit maßgeschneidertem DFÜ-Programm / Reset-Schalter ohne Speicherlöschung. Hardware-Einkaufstips: Drucker, Floppy-Laufwerke, Speichererweiterungen.

Nutzen Sie die Bestellmöglichkeit der Schneider-Sonderhefte 1 und 2 mit der eingehefteten Zahlkarte im vorliegenden Sonderheft von »Happy-Computer«!

Adresse	Bedeutung
AC01	Patch für Ready-Modus
AC04	Patch für Error-Einsprung
AC07	Patch für Befehlsausführung
AC0A	Patch für Funktionsberechnung
AC0D	Patch für Konstante holen (wird durch Interpreterfehler nicht benutzt)
AC10	Patch für Eingabe, Tokenumwandlung
AC13	Patch für Ausgabe, Token listen
AC16	Patch für Ziffernumwandlung bei Eingabe
AC19	Patch für Operatoren

Tabelle 2. Die 9 Patches des Schneider CPC 464

fis geeignet. Um sie zu verstehen, muß man sich den Ablauf der Befehlsanalyse beim Schneider etwas näher ansehen. Wenn der CPC beim Programmlauf oder im Direktmodus auf einen Befehl trifft, so vergleicht er die Zeichenkette, die dieser Befehl für ihn erst einmal darstellt, mit einer Reihe von intern abgespeicherten Zeichenketten, den Basic-Befehlen. In einem zweiten Schritt wird dann, falls ein Befehl ermittelt wurde, der Parameter benötigt – beispielsweise eine Zeilenangabe bei GOTO – und festgestellt, ob überhaupt Parameter vorhanden sind und ob diese den angegebenen Voraussetzungen genügen (Format-Check). Bei einem GOTO-Befehl würde geprüft, ob die angegebene Zeilennummer überhaupt existiert. Nach all diesen Überprüfungen führt der Computer den Befehl aus. Die einzelnen Teile der Interpreterkette sind dabei weitgehend voneinander unabhängig. Die Verbindung zwischen den einzelnen Teilen geschieht mit Hilfe von sogenannten Patches – absichtlichen Flickstellen im System. Bei diesen Patches handelt es sich um einfache RET-Befehle. Der Schneider kehrt also nach dem Aufruf des Patch unmittelbar wieder zu der Position zurück, von der ab der Aufruf erfolgte. Auf den ersten Blick erscheint dies nicht besonders sinnvoll. Bei näherem Hinsehen entpuppt es sich aber als genialer Trick.

Wenn beispielsweise irgendwo ein Fehler enthalten ist – sei es, daß Sie einen Leerraum vergessen haben oder die angegebene Zeilennummer nicht existiert, erfreut Sie der Schneider normalerweise mit seinem reichen Repertoire an Fehlermeldungen. Er verläßt dazu an verschiedenen Stellen die Interpretationskette und ruft einen Programmteil auf, der dann den passenden Kommentar ausgibt. Zwischen Interpreterkette und Fehlerausgabe begibt er sich aber noch zu einer anderen Stelle im RAM, dem sogenannten Error-Patch. Dabei handelt es sich um eine Stelle, die uns Möglichkeit zum Eingreifen bietet.

Da diese Verbindung im RAM gespeichert ist, kann man durch Löschen oder Überschreiben des Patches noch vor der Fehlerausgabe einen eigenen Programmteil einbinden. Dieser muß dann prüfen, ob es sich bei der Fehlermeldung um einen wirklichen Fehler durch Falscheingabe oder vielmehr um einen neuen Befehl handelt, der interpretiert werden soll.

Dieses Verfahren erfordert jedoch einige Sachkenntnis in Hinblick auf die inneren Abläufe des Interpreters. Die Einbindung von Befehlen auf diese Art und Weise ist also nichts für den Anfänger. Wer trotzdem damit experimentieren will, kann die Funktion der einzelnen Patches aus Tabelle 2 entnehmen. Angegeben ist dabei jeweils das erste Byte. Dieses muß durch einen einfachen Sprungbefehl (JP) überschrieben werden. Die Aufrufadresse Ihrer neuen Routine kommt dabei in die nächsten beiden Bytes.

(Carsten Straush/hg)

Massenspeicher besser genutzt



Welchem Massenspeicher Sie auch den Vorzug geben; ob Sie mit Diskette oder Kassette als Speichermedium arbeiten – beide bieten mehr, als Sie vielleicht vermuten.

Mit »IUSER,x«, »IDRIVE«, »IA« und »IB« lassen sich der Benutzerbereich und das verwendete Laufwerk (A oder B) festlegen. Leider gibt es keine Befehle, die erlauben, diese Parameter abzufragen. Mit Hilfe einiger PEEK-Befehle geht es dennoch!

Dazu muß man wissen, daß an der RAM-Adresse BE40 hex die Startadresse des DPH (Disk Parameter Header) von Laufwerk A gespeichert ist. Liest man diese Adresse aus und zieht von ihrem Inhalt den Wert 528 ab, erhält man die Startadresse des von der Floppy belegten RAM-Bereichs:
»START=PEEK(&BE40)+PEEK (&BE41)*256-528«

Am Anfang dieses Speicherbereichs befinden sich die gesuchten Informationen:
»PRINT PEEK(START)« liefert das mit »IDRIVE«, »IA «oder»IB« gewählte Laufwerk.
»PRINT PEEK(START+1)« meldet die aktive User-Nummer (Benutzerbereich).

»PRINT PEEK(START+2)« liefert das zuletzt aktive Laufwerk. Dieser Wert muß nicht unbedingt mit »PEEK(START)« übereinstimmen, denn es gibt ja auch die Möglichkeit, die Laufwerksbezeichnung im Dateinamen anzugeben: beispielsweise ″B:PROG. BIN″

Geben Sie also folgendes ein:
»IA«
»a$=″B:*.*″:IDIR,@a$«
dann resultiert aus »PEEK(START)« eine Null, »PEEK (START+2)« aber der Wert Eins.

Wie Sie die Speicher-Geschwindigkeit des Kassettenrecorders im CPC 464 mit zwei POKE-Befehlen auf rund 3600 Baud erhöhen können, stand in unserem Schneider-Sonderheft Ausgabe 1/86.

Recorder-Tuning

Inzwischen haben wir aber herausgefunden, wie Sie diesen Trick auch auf dem CPC 664 und CPC 6128 anwenden können. Beide Computer besitzen zwar keinen eingebauten Kassettenrecorder, aber statt dessen eine DIN-Buchse zum Anschluß eines externen Recorders.

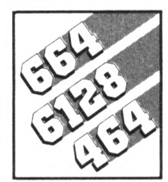
Die Speicherstelle für die Recorder-Geschwindigkeit liegt im CPC 664 und CPC 6128 auf der Adresse B1E9 hex. Die POKE-Befehle für alle drei Schneider-Computer lauten also:

CPC 464: »POKE &B8D1,2:POKE &B8D2,23«

CPC 664 und 6128: »POKE &B1E9,2:POKE &B1EA,23«

(Martin Kotulla/ja)

Serienbriefe – kein Problem



Das Programm WordStar erfreut sich immer größerer Beliebtheit. Die Schar der eifrigen Benutzer sorgt ständig für neue Tips, die seinen Bedienungskomfort vergrößern.

Erst seit kurzem stolzer Besitzer eines Textverarbeitungs-Programms wie Wordstar, brennt man natürlich gleich darauf, dessen fantastische Möglichkeiten zu nutzen. Und da fällt der Blick auf eine Fähigkeit, die wirklich Tolles leistet: MailMerge! Es kann Daten aus einer vorhandenen Datei mit fertigen Texten vermengen und so beispielsweise Serienbriefe selbsttätig drucken und adressieren. Also auf in den Kampf und kurzerhand auf die Adreßdatei zurückgegriffen, die bisher auf irgendeiner Diskette schlummerte.

Alles wunderbar? Denkste! MailMerge mag nämlich Dateien, die unter Amsdos gespeichert wurden, gar nicht so recht leiden. Was nun? Aufgeben und sämtliche Daten eben nochmal neu in eine MailMerge-Datei eingeben? Nein, man schreibt sich eben ein kleines Basic-Programm, das jetzt und in Zukunft die unterschiedlichen Dateiformate entsprechend aufbereitet.

Dazu muß man natürlich wissen, wie beide Formate überhaupt strukturiert sind. Nichts leichter als das. Man lädt unter CP/M die jeweilige Datei in den CP/M-Maschinensprache-Monitor DDT.

Bei der Untersuchung des Speicherinhaltes wird die Struktur schnell deutlich. Nach jedem Datum, das mit »PRINT #9,var$« gespeichert wurde, steht ein Carriage-Return und ein Line Feed (0D0A hex). Diese Byte-Folge findet sich jedoch in einer MailMerge-Datei nur ein einziges Mal, nämlich ausschließlich hinter dem letzten Eintrag.

Die Konsequenz ist, daß das Umwandlungs-Programm die jedem Datenfeld folgenden Feldmarkierungen entfernen und durch ein Komma ersetzen muß.

Nachdem der Name der Amsdos-Datei, die maximale Anzahl der zu lesenden Datensätze, die Anzahl der Datenfelder innerhalb eines Satzes und zuletzt der Name der entstehenden MailMerge-Datei eingegeben sind, liest das Programm die Daten in das Variablenfeld <Adresse$(MaxSatz, MaxFeld)> ein (Programmzeilen 700 bis 860). Dabei wird hinter jedem Datenfeld, den MailMerge-Konventionen entsprechend, ein Komma und ein Leerzeichen gesetzt. Ist die Datei komplett, kann man mit Hilfe des Programm-Blocks in den Zeilen 1300 bis 1410 wählen, welche Daten letztendlich in die neue Datei kommen. Ab der Zeile 1130 beginnt dann die Speicherung der selektierten Daten, so daß Sie zum Beispiel aus einer großen Adreßdatei mehrere verschiedene MailMerge-Dateien generieren können.

Diese Auswahl bestimmter Datensätze funktioniert auch mit MailMerge-Dateien, wenn Sie diese anstelle der Amsdos-Datei laden.

Ein Sonderfall sind Dateien, die außer den eigentlichen Daten noch andere Informationen enthalten, also beispielsweise vor dem ersten Datensatz die Menge der Datensätze. Dann müssen Sie die Datei vor der Umformung erst einmal laden, um sie ohne dieses Feld wieder zu speichern.

Nach vollzogener Umwandlung legen Sie unter WordStar die WS-Mix-Datei an, zum Beispiel durch:

```
.op
.df Name.dat
.rv vname,nname,str,ort,tel
```

Selbstverständlich muß die hier angegebene Zahl der Datenfelder (in diesem Falle fünf) mit der tatsächlichen Anzahl in der Datei übereinstimmen. Ansonsten verfahren Sie, wie es im Wordstar-Handbuch, Anhang C1, beschrieben ist.

(H.J.Hesse/ja)

```
10    '*********************************** [6BA4]
20    '***    KONVERTIEREN EINER       *** [D768]
25    '***                             *** [7094]
30    '*** >> SCHNEIDER  CPC  X64 << *** [450E]
35    '***                             *** [A496]
40    '***  AMSDOS - ADRESS - DATEI    *** [8262]
45    '***                             *** [E498]
50    '***         IN  EINE            *** [9B80]
60    '***                             *** [CB92]
70    '** WORDSTAR - MAILMERG - DATEI ** [0CB6]
80    '***             ========        *** [4566]
90    '***                             *** [2D98]
95    '*********************************** [E6BE]
100   '***                             *** [CDA8]
110   '***    ( C )   1985  by         *** [0FF6]
115   '***                             *** [3CB4]
120   '***    Hans-Joachim Hesse       *** [3040]
130   '***                             *** [CEAE]
140   '***    2940 Wilhelmshaven       *** [63BC]
150   '***                             *** [30B2]
160   '***         Tel: 04421/305137 *** [E42C]
170   '*********************************** [00D2]
180   '                                    [0100]
190 GOTO 1500      ' HauptProgramm         [B0CE]
200   '================================    [A4AC]
210   '     BILDSCHIRM FORMATIEREN         [316E]
220   '================================    [F6B0]
225 MODE 2                                 [A460]
230  WINDOW #0,1,80,4,22                   [1402]
235 PAPER #0,0:PEN #0,1                    [0FDC]
240  WINDOW #1,1,80,1,3                    [F99E]
245 PAPER #1,1:PEN #1,0                    [94E2]
250  WINDOW #2,1,80,23,25                  [F672]
255 PAPER #2,0:PEN #2,1                    [53E8]
260 FOR I = 0 TO 2:CLS #(I): NEXT I        [B18A]
270 RETURN                                 [A932]
300  '================================     [EEAE]
310  '     BILDSCHIRM - AUFBAU             [A0FA]
320  '================================     [2CB2]
330 CLS #1:LOCATE #1,10,2                  [3ABA]
340 PRINT #1,CHR$(24);"{14 SPACE}";        [192C]
341 PRINT #1,"AMSDOS - DATEI{3 SPACE}KON
    VERTIEREN{11 SPACE}";                  [32B6]
342 PRINT #1,CHR$(24)                      [F83C]
390 RETURN                                 [AA38]
400  '================================     [F8B0]
410  '     PARAMETER - EINGABE             [55EA]
420  '================================     [A6B4]
430  CLS : LOCATE 3,5                      [FE3E]
440 INPUT "Name der AMSDOS-DATEI{8 SPACE
    }?{2 SPACE}",AmsDat$                   [2A18]
450 LOCATE  3,9                            [6E92]
460 INPUT "WIEVIELE{3 SPACE}DATENSATZE{3
     SPACE}MAX{2 SPACE}?{2 SPACE}",MaxSa
    tz                                     [E8A8]
470 LOCATE 3,11                            [98A8]
480 INPUT "WIEVIELE{3 SPACE}Datenfelder{
    2 SPACE}MAX{2 SPACE}?{2 SPACE}",MaxF
    eld                                    [9DF4]
490 LOCATE 3,15                            [32B4]
500 INPUT "Name der MailMerge - Datei{3
    SPACE}?{2 SPACE}",MailDat$             [E56E]
510 CLS #2:LOCATE #2,15,2                  [82C8]
520 INPUT #2,"EINGABE KORREKT{2 SPACE}(J
    /N) ";ja$                              [5DF6]
530 IF UPPER$(ja$)="J" THEN 550 ELSE 540
                                           [1A5A]
540 CLS #2: GOTO 430                       [15B4]
550  RETURN                                [BE74]
```

Listing. Basic-Dateien mit Wordstar nutzen

```
600  '=================================  [3CB4]
610  '     VARIABLEN + KONSTANTEN          [6404]
620  '=================================  [8EB8]
630 DIM Adresse$(MaxSatz,MaxFeld)       [D408]
640 sz = 0    '  SatzZaehler             [9010]
650 fb = 1    '  FeldNummer              [B1CE]
660 sn = 1    '  SatzNummer              [EE50]
670 RETURN                               [7D3A]
700 '=================================   [A376]
710 '     AMSDOS - DATEI  LESEN          [9B62]
720 '=================================   [537A]
730 CLS : CLS #2:LOCATE 24,10            [CAE4]
740 PRINT " Einen Moment bitte"          [F606]
750 LOCATE  24,8                         [89FC]
760 PRINT "Die AMSDOS-DATEI wird geladen
    "                                    [EE02]
770 OPENIN AmsDat$                       [A90A]
780 WHILE NOT EOF                        [3786]
790 INPUT #9,a$                          [0CFA]
800 a$=MID$(a$,1)+", "                   [93CC]
810 Adresse$(sn,fb)=a$                   [9618]
820 fb=fb+1                              [24C6]
830 IF fb=MaxFeld+1 THEN sn=sn+1         [466C]
840 IF fb=MaxFeld+1 THEN fb=1            [2324]
850 WEND                                 [6DD6]
860 CLOSEIN                              [5D96]
870 RETURN                               [A33E]
900 '=================================   [E57A]
910 '    AMSDOS-DATEI  KONVERTIEREN      [5430]
920 '=================================   [357E]
930 DIM Adr$(sn)                         [9046]
940 Satz=1                               [049A]
950 FOR Feld = 1 TO MaxFeld              [E0E4]
960 IF Satz = sn THEN RETURN             [617A]
970 '                                    [73CE]
980 Adr$(Satz)=Adr$(Satz)+Adresse$(Satz,
    Feld)                                [9214]
990 NEXT Feld                            [BC38]
1000 Adr$(Satz)=LEFT$(Adr$(Satz),LEN(Adr
     $(Satz))-2)                         [F306]
1010 GOTO 1300                           [8CFE]
1020 Satz=Satz+1                         [F480]
1030 GOTO 950                            [68B6]
1100 '=================================  [0DCC]
1110 '     ABSPEICHERN  MAILMERG.ADR     [8682]
1120 '=================================  [5DD0]
1130 OPENOUT MailDat$                    [36DE]
1140 FOR i = 1 TO sn-1                   [374C]
1150 IF Adr$(i) = "/*" THEN 1170         [A9BA]
1160 sz=sz+1: PRINT #9,Adr$(i)           [D5BE]
1170 NEXT i                              [A462]
1180 CLOSEOUT                            [44B0]
1190 CLS: MODE 2                         [C53C]
1200 LOCATE 20,10                        [3A54]
1210 PRINT "Die{2 SPACE}MailMerge-Datei{
     2 SPACE}ist geladen"                [FD5C]
1220 LOCATE 24,13                        [7C66]
1230 PRINT "mit{2 SPACE}> ";sz;" <{3 SPA
     CE}Saetzen."                        [AA54]
1240 PRINT:PRINT:PRINT:PRINT:END         [DBB4]
1300 '=================================  [5FD0]
1310 '     AUSWAHL AUS AMSDOS-ADRESSEN   [1526]
1320 '=================================  [0FD4]
1330 CLS:PRINT CHR$(24);                 [7546]
1340 PRINT TAB(14)," A U S W A H L"+STRI
     NG$(40,32);CHR$(24)                 [CACC]
1350 LOCATE 10,10:PRINT Adr$(satz)       [08C8]
1360 LOCATE 10,15:PRINT"Soll dieser Satz
     ";                                  [5126]
1370 PRINT"in die MailMerge-Datei ?"     [09D6]
1380 LOCATE 24,18                        [A17E]
1390 INPUT "Bitte{2 SPACE}waehlen{2 SPAC
     E}Sie{3 SPACE}< J/N >{2 SPACE}",ja$
                                         [AF34]
1400 IF UPPER$(ja$)="J" THEN GOTO 1020   [0FB4]
1410 Adr$(Satz)="/*": GOTO 1020          [6BC8]
1420 '                                   [971C]
1500 '=================================  [91D4]
1510 '     HAUPTSTEUERUNG                [9EA4]
1520 '=================================  [E1D8]
1530 GOSUB 200    ' WINDOW'S DECLARIEREN  [34A4]
1540 GOSUB 300    ' BILDSCHIRM-AUFBAU     [C90C]
1550 GOSUB 400    ' PARAMETER ABFRAGE     [C0DE]
1560 GOSUB 600    ' VARIABLEN+KONSTANTE   [907E]
1570 GOSUB 700    ' AMSDOS-DATEI LESEN    [737A]
1580 GOSUB 900    ' AMSDOS-DAT KONVERTIE  [A8E4]
1590 GOSUB 1100   ' MAILMERG-DATEI SPEIC  [349C]
```

Listing. Basic-Dateien mit Wordstar nutzen (Schluß)

Information total



Wer hätte gedacht, daß es über Diskettendateien viel mehr zu erfahren gibt als das, was DIR- oder CAT-Befehl ohnehin schon anzeigen? Lassen Sie sich überraschen.

Warum soll es eigentlich nur für Kassettenbetrieb sogenannte Header-Lese-Programme geben? Dieses Listing zeigt, daß es auch mit Diskette ohne weiteres möglich ist.

Der »Header« (Programm-Kopf) enthält Informationen, die der Computer benötigt, um Programme beziehungsweise Dateien richtig laden zu können. Dort steht beispielsweise, ob es sich um ein Basic- oder Maschinensprache-Programm handelt, an welche Adresse es zu laden ist und wo die Startadresse liegt. So etwas ist natürlich auch für den Benutzer von Interesse.

Nach dem Start müssen Laufwerksnummer und Diskettenformat angegeben werden. Danach erscheinen sämtliche Dateinamen am linken Bildschirmrand. Neben den Namen finden Sie die Angaben über Ladeadresse, exakte Länge in Byte, Einsprungadresse (falls vorhanden), File-Attribute (R/W, R/O, etc.) und User (Benutzer-Bereich).

Das Programm ist für den CPC 464 mit Schneider-Diskettenlaufwerk DDI-1 konzipiert. Durch entsprechende Änderungen läßt es sich aber sicher auch an andere Laufwerke anpassen.

(D.Babirat/ja)

```
10 REM ***********************************
   ********************                  [8F6E]
20 REM *
                       *                 [AC74]
30 REM *        HEADERANALYSE VON DISKETT
   ENPROGRAMMEN        *                 [D04E]
40 REM *----------------------------------
   -------------------*                  [07A6]
50 REM *    D.BABIRAT / STERNSTR.16 / 230
   0 KIEL 1 / 10.85    *                 [FE84]
60 REM *             alle Rechte bei obi
   gem Autor           *                 [89CA]
70 REM ***********************************
   ********************                  [A57A]
80 MODE 2:MEMORY &8000                   [2040]
90 REM                                   [40DA]
100 REM ****** LESEPROGRAMM EINPOKEN ***
    ***                                  [8FE8]
110 REM                                  [492C]
120 DATA &21,&1C,&50,&CD,&D4,&BC,&22,&1D
                                         [43D4]
130 DATA &50,&79,&32,&1F,&50,&21,&20,&50
                                         [E724]
140 DATA &5E,&23,&56,&23,&4E,&21,&30,&50
                                         [A74C]
150 DATA &DF,&1D,&50,&C0,&84             [9A42]
160 FOR adr=&5000 TO &501C:READ byte:POK
    E ADR,BYTE:NEXT ADR                  [4BB2]
170 GOSUB 1300 : REM **** SYMBOLDEFINITI
    ON ****                              [AFD2]
180 REM                                  [763A]
190 REM *** MENUE ***                    [91E8]
200 REM                                  [602C]
210 LOCATE 23,25:PRINT "DISCHEADERANALYS
    E by ";CHR$(241)CHR$(242)CHR$(243)CH
    R$(244)"-"CHR$(245)CHR$(246)CHR$(247
    )CHR$(248)                           [9064]
220 LOCATE 23,23:PRINT"Diskette einlegen
     und Taste druecken !":CALL &BB06    [CAE2]
```

Listing. Voller Durchblick beim Disketten-Inhalt

```
230 LOCATE 23,23:PRINT STRING$(55,32)       [1862]
240 LOCATE 23,23:INPUT"WELCHES LAUFWERK
    (A/B)";DRIVE$                           [0134]
250 TYPE$="":LOCATE 23,23:INPUT"CP/M ode
    r DATEN-Diskette (C/D)";TYPE$           [C496]
260 LOCATE 23,23:PRINT STRING$(55,32)       [5568]
270 REM                                     [5F3A]
280 REM *** BILDSCHIRM VORBEREITEN ***      [3780]
290 REM                                     [3D3E]
300 WINDOW#1,2,79,4,19:CLS#1                [3C7E]
310 PLOT 1,7:DRAW 1,368:PLOT 1,368:DRAW
    639,368:PLOT 639,368:DRAW 639,7:PLOT
    639,7:DRAW 445,7:PLOT 445,7:DRAW 44
    5,20:PLOT 445,20:DRAW 160,20:PLOT 16
    0,20:DRAW 160,7:PLOT 160,7:DRAW 1,7     [8944]
320 LOCATE 3,2:PRINT"NAME":LOCATE 19,2:P
    RINT"ATTRIBUT":LOCATE 33,2:PRINT"STA
    RT":LOCATE 44,2:PRINT"LAENGE":LOCATE
    56,2:PRINT"EINSPRUNG":LOCATE 70,2:P
    RINT"USER"                              [8EBC]
330 ZEILE=4:NUMBER=0:INK 0,17:INK 1,0:BO
    RDER 17                                 [D7B2]
340 REM                                     [6236]
350 REM **** SCHLEIFE FUER MAX. 16 EINTR
    AEGE PRO SEKTOR ****                    [23B2]
360 REM                                     [643A]
370 IF UPPER$(DRIVE$)="B" THEN DRIVE=1 E
    LSE DRIVE=0                             [FFE4]
380 IF UPPER$(TYPE$)<>"D" THEN 400          [45AC]
390 DIRTRACK=0:DIRSECTOR=1:OFFSET=192:GO
    TO 410      '**** DATEN-FORMAT ****     [121E]
400 DIRTRACK=2:DIRSECTOR=1:OFFSET=64
                '**** CP/M FORMAT  ****     [F04A]
410 POKE &5020,DRIVE:POKE &5021,DIRTRACK
    :POKE &5022,DIRSECTOR+OFFSET            [59A6]
420 BUFFER=&5030-32:CALL &5000              [4C1C]
430 LOOP=1:LOCATE 3,22:PRINT"GELESENER S
    ECTOR :&";:PRINT HEX$(DIRSECTOR+OFFS
    ET)                                     [4ACC]
440 BUFFER=BUFFER+32:IF BUFFER>&522F THE
    N 790                                   [8DDC]
450 REM                                     [613A]
460 REM **** ATTRIBUTE UND ERA UNTERSUCH
    EN UND MERKEN ****                      [6BBC]
470 REM                                     [633E]
480 IF PEEK(BUFFER+12)<>0 THEN 700
        '*** NOT FIRST BLOCK ***            [F6C0]
490 IF PEEK(BUFFER)=&E5 THEN 700
        '*** ERA ***                        [4EF4]
500 USER=PEEK(BUFFER)
        '*** USER **                        [4BF2]
510 IF PEEK(BUFFER+15)=&80 THEN MULTI=1
        '*** >16K **                        [EEC4]
520 IF PEEK(BUFFER+9)>127 THEN RO=1
        '*** R/O ***                        [E3C6]
530 IF PEEK(BUFFER+10)>127 THEN SYS=1
        '*** SYS ***                        [AFB2]
540 REM                                     [623A]
550 REM **** NAMEN MIT EXTENSION AUS BUF
    FER LESEN ****                          [4B3C]
560 REM                                     [603E]
570 FOR X=BUFFER+1 TO BUFFER+8:NAME$=NAM
    E$+CHR$(PEEK(X)):NEXT                   [1766]
580 FOR X=BUFFER+9 TO BUFFER+11:EXTENSIO
    N$=EXTENSION$+CHR$(PEEK(X)):NEXT        [FF3C]
590 LOCATE 3,ZEILE:PRINT NAME$;:PRINT"."
    ;:PRINT EXTENSION$;                     [4E98]
600 NUMBER=NUMBER+1:LOCATE 3,23:PRINT"GE
    SAMTEINTRAEGE:";:PRINT NUMBER           [2878]
610 IF RIGHT$(EXTENSION$,1)=CHR$(77) THE
    N COM=1                                 [40A0]
620 IF RO=1 THEN LOCATE 19,ZEILE:PRINT"$
    R/O-";:RO=0                             [06F6]
630 IF SYS=1 THEN LOCATE 24,ZEILE:PRINT"
    SYS";:SYS=0                             [2224]
640 IF COM=1 THEN 750                       [B582]
650 BLOCK=PEEK(BUFFER+16):GOSUB 830
     REM **** ZUR DATENAUSWERTUNG ****
                                            [09A0]
660 IF MULTI=1 THEN 680
     REM **** FILE LAENGER ALS 16K **       [3982]
670 LOOP=LOOP+1:ZEILE=ZEILE+1:IF LOOP<17
    THEN 440                                [82BA]
680 MULTI=0:BUFFER=BUFFER+32:IF BUFFER>&
    522F THEN 790                           [C64C]
690 LOOP=LOOP+1:ZEILE=ZEILE+1:IF LOOP<17
    THEN 440                                [C0BE]
700 LOOP=LOOP+1:IF LOOP<17 THEN 710         [6A24]
710 IF BUFFER>&522F THEN 790 ELSE 440       [41E8]
720 REM                                     [683A]
730 REM **** VARIABLEN ZURUECKSETZEN ***
    *                                       [2F08]
740 REM                                     [623E]
750 NAME$="":EXTENSION$="":COM=0:ZEILE=Z
    EILE+1:LOOP=LOOP+1                      [D960]
760 LOCATE 70,ZEILE-1:PRINT USER            [B14A]
770 IF LOOP<17 THEN 780 ELSE 790            [1BC4]
780 IF BUFFER>&52FF THEN 790 ELSE 440       [3D1E]
790 LOCATE 27,23:PRINT"WEITER MIT <TASTE
    >":CALL &BB06                           [EA30]
800 LOCATE 27,23:PRINT STRING$(20,32):CL
    S#1:ZEILE=4                             [EB88]
810 DIRSECTOR=DIRSECTOR+1:IF DIRSECTOR<5
    THEN 410 ELSE 190                       [B880]
820 REM                                     [AA3C]
830 REM **** BLOCKNUMMER IN TRACK UND SE
    CTOR UMRECHNEN ****                     [0C2C]
840 REM                                     [0C40]
850 A=BLOCK                                 [EC4C]
860 XH=FIX(A/9):YH=A MOD 9                  [1B5C]
870 IF YH<5 THEN F=0 ELSE F=1               [017E]
880 TRACK=XH*2+2+F:F=0                      [6952]
890 IF UPPER$(TYPE$)="D" THEN TRACK=TRAC
    K-2                                     [ED22]
900 SECTOR=(A*2+18) MOD 9 +1                [9C3A]
910 IF TRACK>42 THEN 1250                   [F572]
920 IF SECTOR>9 THEN 1250                   [E5D0]
930 REM                                     [B740]
940 REM *** TRACK UND SECTOR FUER FILEDA
    TEN LESEN ***                           [4EFC]
950 REM                                     [7D44]
960 POKE &5020,DRIVE:POKE &5021,TRACK:PO
    KE &5022,SECTOR+OFFSET:CALL &5000       [EA00]
970 BUFFER2=&5030                           [8FAE]
980 START=PEEK(BUFFER2+22)*256+PEEK(BUFF
    ER2+21)                                 [27BE]
990 LAENGE=PEEK(BUFFER2+25)*256+PEEK(BUF
    FER2+24)                                [7708]
1000 ENTRY=PEEK(BUFFER2+28)*256+PEEK(BUF
     FER2+27)                               [E01E]
1010 REM                                    [2D8C]
1020 REM **** FILEDATEN AUF BILDSCHIRM A
     USGEBEN ****                           [51D4]
1030 REM                                    [2F90]
1040 LOCATE 33,ZEILE:PRINT"&";:IF START<
     4096 THEN PRINT"0"+HEX$(START):GOTO
     1070                                   [2A30]
1050 PRINT HEX$(START)                      [11F6]
1060 REM ****                               [0926]
1070 LOCATE 44,ZEILE:PRINT"&";:IF LAENGE
     <16 THEN PRINT"000"+HEX$(LAENGE):GO
     TO 1120                                [8F92]
1080 IF LAENGE<256 THEN PRINT"00"+HEX$(L
     AENGE):GOTO 1120                       [B78A]
1090 IF LAENGE<4096 THEN PRINT"0"+HEX$(L
     AENGE):GOTO 1120                       [1998]
1100 PRINT HEX$(LAENGE)                     [F72A]
1110 REM ****                               [A01E]
1120 LOCATE 56,ZEILE:PRINT"&";:IF ENTRY<
     16 THEN PRINT"000"+HEX$(ENTRY):GOTO
     1170                                   [B232]
1130 IF ENTRY<256 THEN PRINT"00"+HEX$(EN
     TRY):GOTO 1170                         [0524]
1140 IF ENTRY<4096 THEN PRINT"0"+HEX$(EN
     TRY):GOTO 1170                         [AD32]
1150 PRINT HEX$(ENTRY)                      [2C00]
1160 REM ****                               [0628]
1170 LOCATE 70,ZEILE:PRINT USER             [CCE6]
1180 NAME$="":EXTENSION$="":RO=0:SYS=0      [BC74]
1190 REM                                    [719E]
1200 REM **** DIRECTORY EINLESEN UND WEI
     TER AUSWERTEN ****                     [84E8]
1210 REM                                    [3590]
1220 POKE &5020,DRIVE:POKE &5021,DIRTRAC
     K:POKE &5022,DIRSECTOR+OFFSET:CALL
     &5000                                  [0BC8]
1230 RETURN                                 [C98C]
1240 REM                                    [1C96]
1250 REM *** HARDWAREFEHLER ***             [6F58]
1260 REM                                    [2A9A]
1270 LOCATE 20,24:PRINT"Falsche Track od
     er Sectornummer -- Taste druecken "
                                            [324C]
1280 CALL &BB06:LOCATE 20,24:PRINT STRIN
     G$(60,32):GOTO 1180                    [6688]
1290 REM                                    [0DA0]
1300 REM **** SYMBOLDEFINITION ****         [1B6E]
1310 REM                                    [2992]
1320 SYMBOL 241,0,63,33,33,32,33,33,63
     : REM C                                [4166]
1330 SYMBOL 242,0,33,33,33,33,33,33,63
     : REM U                                [598A]
1340 SYMBOL 243,0,62,34,34,63,33,33,63
     : REM B                                [7C76]
1350 SYMBOL 244,0,62,32,32,60,32,32,62
     : REM E                                [BA6C]
1360 SYMBOL 245,0,62,34,32,62,2,34,62
     : REM S                                [2172]
1370 SYMBOL 246,0,62,34,34,34,34,34,62
     : REM O                                [499A]
1380 SYMBOL 247,0,62,32,32,60,32,32,32
     : REM F                                [9874]
1390 SYMBOL 248,0,63,12,12,12,12,12,12
     : REM T                                [A27C]
1400 RETURN                                 [AD8A]
```

Listing. Voller Durchblick beim Disketteninhalt (Schluß)



Geschwindigkeit ist keine Hexerei



Wahrscheinlich haben Sie sich auch schon einmal darüber geärgert. Das Formatieren mit dem Schneider-Laufwerk dauert lange. Format-Plus schafft endlich Abhilfe.

Völlig ohne CP/M können Sie nun in Windeseile Disketten formatieren. Das Programm ist für den Controller des Diskettenlaufwerks zum CPC 464 ausgelegt, arbeitet aber auch auf dem CPC 664 und 6128 (dann funktioniert lediglich die Anzeige des aktuell formatierten Sektors auf dem Bildschirm nicht). Besonders interessant ist, daß damit auch in einem angeschlossenen Zweitlaufwerk formatiert werden kann, was sonst ja nicht machbar ist. Ein Durchgang benötigt so nur noch zirka 15 Sekunden anstatt der bislang üblichen 37 Sekunden, allerdings unter Verzicht auf die automatische Fehlerprüfung. Wer will, kann die Maschinencode-Routine in eigene Programme einbauen. Erzeugt wird wahlweise CP/M-(Vendor-) oder Datenformat.

(D. Babirat/ja)

```
10 REM ************************************
   **********************                [666A]
20 REM * PROGRAMM ZUM DIREKTEN FORMATIER
   EN AUF DRIVE A ODER B *               [2220]
30 REM * COPYRIGHT by D.Babirat / 2300 K
   iel 1 / Sternstr. 16  *               [A7D2]
40 REM *                    geschrieben 1
   0.85                    *             [022E]
50 REM ************************************
   **********************                [BE72]
60 REM                                   [FFD4]
70 REM                                   [0AD6]
80 REM **** LOGO AUFBAUEN ****           [A768]
90 REM                                   [40DA]
100 GOSUB 590                            [63DE]
110 LOCATE 15,4:PRINT"'FORMAT' by "CHR$(
    241)CHR$(242)CHR$(243)CHR$(244)"-"CH
    R$(245)CHR$(246)CHR$(247)CHR$(248)   [DBC4]
120 PLOT 1,1:DRAW 1,300:PLOT 1,300:DRAW
    639,300:PLOT 639,300:DRAW 639,1:PLOT
     639,1:DRAW 1,1                      [9D72]
130 PLOT 100,330:DRAW 100,356:PLOT 100,3
    56:DRAW 300,356:PLOT 300,356:DRAW 30
    0,330:PLOT 300,330:DRAW 100,330      [E67C]
140 REM                                  [6232]
150 REM **** MASCHINENPROGRAMM POKEN ***
    *                                    [ACC4]
160 REM                                  [6036]
170 GOSUB 750                            [5AE8]
180 REM                                  [763A]
190 REM *** DATEN ABFRAGEN ***           [B878]
200 REM                                  [602C]
210 CLS#1:LOCATE 10,10:INPUT"AUF WELCHEM
     LAUFWERK FORMATIEREN (A/B)";DRIVE$  [ECA2]
220 IF UPPER$(DRIVE$)="B" THEN DRIVE=1 E
    LSE DRIVE=0                          [DAD8]
230 LOCATE 10,12:INPUT"DATEN ODER CP/M-F
    ORMAT (D/C)";FORMAT$                 [3F6A]
240 IF UPPER$(FORMAT$)="D" THEN ZEROSEC=
    &C1 ELSE ZEROSEC=&41                 [638A]
250 REM                                  [6136]
260 REM **** DATEN AN PROGRAMM UEBERGEBE
    N ****                               [89C0]
270 REM                                  [5F3A]
280 POKE &4078,DRIVE                     [CE50]
290 ZEROTRACK=0:POKE &4079,ZEROTRACK     [B882]
300 POKE &407A,ZEROSEC                   [DA96]
310 REM                                  [5930]
320 REM **** DATEN IN TABELLE FUER FDC U
    EBERTRAGEN ****                      [4630]
330 REM                                  [4B34]
340 ADR=&4098                            [D68C]
350 FOR LOOP=1 TO 9                      [2B06]
360 POKE ADR,DRIVE                       [A90A]
370 POKE ADR+1,0                         [3930]
380 POKE ADR+2,ZEROSEC                   [450A]
390 POKE ADR+3,2                         [553C]
400 ADR=ADR+4                            [0EFC]
410 ZEROSEC=ZEROSEC+2:IF (ZEROSEC AND &F
    )=&B THEN ZEROSEC=ZEROSEC-9          [0C8C]
420 NEXT                                 [D2EA]
430 REM                                  [6736]
440 REM **** DISKETTE ANFORDERN ****     [3DD0]
450 REM                                  [613A]
460 LOCATE 10,14                         [2408]
470 PRINT"BITTE ZU FORMATIERENDE DISKETT
    E EINLEGEN UND TASTE DRUECKEN !"     [64EC]
480 CALL &BB06                           [C110]
490 LOCATE 10,14:PRINT STRING$(68,32)    [D172]
500 CALL &4000                           [61B6]
510 REM                                  [6134]
520 REM **** ENDE UND RUECKSPRUNG ****   [25CE]
530 REM                                  [6F38]
540 IF FORMAT$="D" THEN 560 ELSE 550     [524E]
550 CLS#1:LOCATE 10,10:PRINT"DIESE CP/M-
    DISKETTE ENTHAELT KEINE BESCHRIEBENE
    N SYSTEMSPUREN !"                    [AD1E]
560 LOCATE 10,14:PRINT"FORMATIERUNG BEEN
    DET! NOCH EINE DISKETTE FORMATIEREN
    (J/N) ";                             [23A0]
570 INPUT TES$ :IF UPPER$(TES$)="J" THEN
     190 ELSE CALL 0                     [447E]
580 REM                                  [3E42]
590 REM **** SYMBOLDEFINITION ****       [9722]
600 REM                                  [A834]
610 MODE 2:INK 0,20:BORDER 20:INK 1,0    [29D6]
620 WINDOW#1,3,77,10,23                  [20F6]
630 SYMBOL 241,0,63,33,33,32,33,33,63  :
     REM C                               [C60C]
640 SYMBOL 242,0,33,33,33,33,33,33,63  :
     REM U                               [FE30]
650 SYMBOL 243,0,62,34,34,63,33,33,63  :
     REM B                               [451C]
660 SYMBOL 244,0,62,32,32,60,32,32,62  :
     REM E                               [BF12]
670 SYMBOL 245,0,62,34,32,62,2,34,62   :
     REM S                               [2618]
680 SYMBOL 246,0,62,34,34,34,34,34,62  :
     REM O                               [6E40]
690 SYMBOL 247,0,62,32,32,60,32,32,32  :
     REM F                               [851A]
700 SYMBOL 248,0,63,12,12,12,12,12,12  :
     REM T                               [3810]
710 RETURN                               [CE30]
720 REM                                  [683A]
730 REM **** DATEN FUER FORMATPROGRAMM *
    ***                                  [2CB4]
740 REM                                  [623E]
750 DATA &F5,&C5,&D5,&E5,&21,&37,&C3,&22
    ,&7B,&40                             [13D0]
760 DATA &3E,&FD,&32,&7D,&40,&21,&79,&EE
    ,&22,&92                             [DEE2]
770 DATA &40,&3E,&FD,&32,&94,&40,&21,&52
    ,&C6,&22                             [E786]
780 DATA &95,&40,&3E,&07,&32,&97,&40,&26
    ,&0A,&2E                             [8076]
790 DATA &0E,&CD,&75,&BB,&21,&7E,&40,&DF
    ,&7B,&40                             [2930]
800 DATA &C3,&5F,&40,&3A,&79,&40,&57,&3A
    ,&78,&40                             [4A94]
810 DATA &5F,&3A,&7A,&40,&4F,&21,&98,&40
    ,&DF,&95                             [B0E0]
820 DATA &40,&3A,&79,&40,&3C,&FE,&28,&CA
    ,&73,&40                             [88CA]
830 DATA &32,&79,&40,&21,&98,&40,&06,&09
    ,&77,&23                             [A320]
840 DATA &23,&23,&23,&10,&F9,&26,&21,&2E
    ,&0E,&CD                             [4F96]
850 DATA &75,&BB,&21,&00,&00,&3A,&79,&40
    ,&6F,&DF                             [96C0]
860 DATA &92,&40,&C3,&35,&40,&E1,&D1,&C1
    ,&F1,&C9                             [67C6]
870 DATA &00,&00,&00,&00,&00,&00,&46,&4F
    ,&52,&4D                             [F008]
880 DATA &41,&54,&49,&45,&52,&45,&20,&54
    ,&52,&41                             [291A]
890 DATA &43,&4B,&20,&3A,&20,&00,&00,&00
    ,&00,&00                             [A7F0]
900 DATA &00,&00,&00,&00                 [9F1E]
910 FOR adr= 16384  TO 16537             [ECC8]
920 READ BYTE:POKE ADR,BYTE:NEXT         [46C8]
930 RETURN                               [B038]
```

Listing. Beschleunigung durch »Format-Plus«

Faszination Leben



Mathematik muß nicht trocken sein. Das Spiel »Leben« zeigt, wie logische Regeln Symbole in Bewegung versetzen. Zweidimensionale »Computer«, elektronische Gatter oder ganze Welten werden mit »Life« simuliert.

Für viele ist die Mathematik eine grausam trockene Wissenschaft. Ein treffendes Gegenbeispiel liefert ein Spiel, das einem der höheren Gebiete dieser »reinsten aller Wissenschaften« entstammt. Es wurde um 1970 von dem britischen Mathematiker John Horton Conway an der Universität von Cambridge entwickelt und auf den Namen »Leben« (Life) getauft. »Leben« ist ein Spiel für einen Spieler, oder besser Zuschauer, wenn man es auf dem Computer spielt. Die Regeln sind leicht zu erlernen. Probiert man es jedoch auf einem Damebrett, was ebenfalls möglich ist, muß man schon höllisch aufpassen. Einigen von Ihnen dürfte diese Kurzweil bereits bekannt sein. Für sie sind der Abschnitt über Varianten sowie das Programmschema gedacht.

Basis des Spiels ist eine beliebig große Ebene mit schachbrettgleicher Einteilung. Man kann, um die Regeln einzuüben, einfach ein Blatt kariertes Papier und einen Bleistift benutzen. Ein Karo entspricht jeweils einer Zelle. Später wollen wir uns dann die Abläufe überlegen, die man für ein entsprechendes Computerprogramm benötigt.

Bei Spielbeginn werden einige in der Regel zusammenhängende Zellen zum »Leben« erweckt und dementsprechend gekennzeichnet. Demnach hat jede Zelle entweder den Zustand tot (0) oder lebendig (1). Danach ist die Kreativität des Spielers nicht mehr gefragt, denn ab sofort übernimmt »Leben« selbst die Regie über die Fortentwicklung der Anfangskonstellation. Im Kosmos des »Lebens«, den wir im folgenden Mikronesien nennen wollen (seine Bewohner nennen sich Miks), läuft die Zeit in diskreten Takten ab, den Mikronen. Von einer Mikrone zur nächsten werden folgende Änderungen vorgenommen:

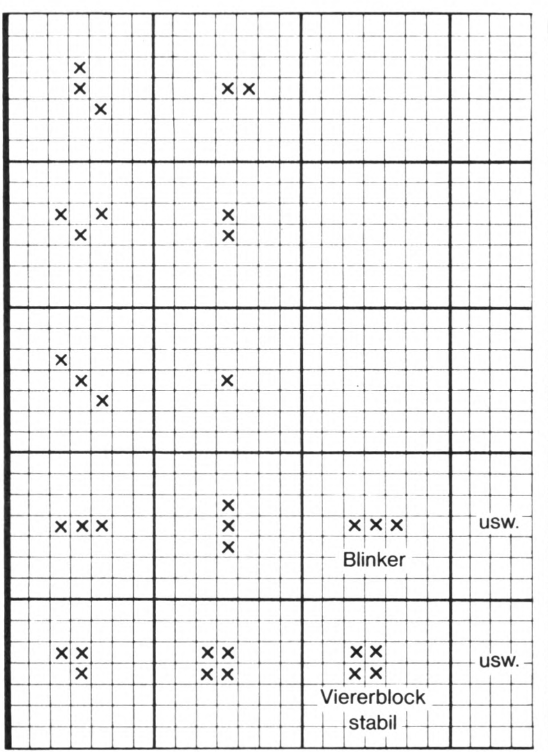
Leben gehorcht einfachen Regeln

1. Geboren wird ein Mik nur in einer Zelle, die genau drei Nachbarn aufweist.
2. Einerseits stirbt Mik an Einsamkeit, wenn sich in seinen insgesamt acht Nachbarfeldern weniger als zwei Kameraden befinden. Andererseits stirbt er einen Erstickungstod, wenn in seiner Umgebung mehr als drei Miks wohnen.
3. Der Mik überlebt nur, wenn seine Zelle entweder zwei oder drei belebte Nachbarzellen besitzt.
4. Ein Mik kann sich niemals in eine andere Zelle bewegen.

Das sind schon alle Grundregeln. Zum sicheren Verständnis dieser »Lebensregeln« wollen wir einmal alle möglichen Minimal-Populationen analysieren, die sich mit maximal vier Miks bilden lassen (Bild 1). Bei der Überprüfung wird folgendermaßen verfahren:

Eine belebte Zelle wird zunächst auf ihre Zukunftsperspektiven hin (Überleben oder Sterben) geprüft. Anschließend werden die unbelebten Nachbarzellen des gleichen Miks auf eine bevorstehende Geburt hin überprüft. Wird eine Änderung erkannt, so müssen die entsprechenden Zellen gekennzeichnet werden. Auf diese Weise wird jeder Mik getestet. Erst nachdem sämtliche Kennzeichnungen vorgenommen wurden, erfolgt der Sprung in die nächste Mikrone, und die Miks werden entfernt oder neu gesetzt.

Bei der Betrachtung von Bild 1 wird sofort klar, daß Zweiergruppen von Miks sich in der nächsten Mikrone in nichts auflösen (Vereinsamung). Ebenso reizlos sind die Kombinationen



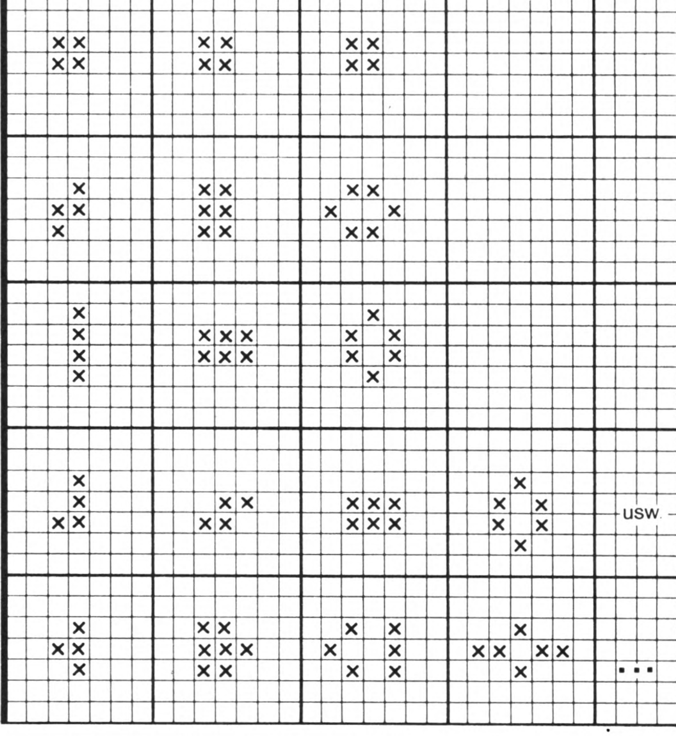


Bild 1. Mit drei oder vier »Miks« kann man jeweils fünf verschiedene Kombinationen bilden (jeweils erste Spalte)

aus nur drei Miks. Drei der möglichen Kombinationen verschwinden schon nach der zweiten Mikrone, ein Drei-Mik geht in einen stabilen Viererblock über, der fünfte Drei-Mik schließlich ist ein sogenannter Blinker (Bild 1). Bei den Vier-Miks sind ebenfalls fünf verschiedene Anfangsfiguren möglich. Vier enden stabil, ein Vier-Mik bildet nach neun Mikronen vier Drei-Mik-Blinker. Spannend wird es erst bei den Fünf-Miks. Das sogenannte »r-Pentomino« (Bild 2) ist die kleinste Figur, die sich (mit großer Wahrscheinlichkeit) über alle Grenzen hinaus ausdehnt. Es zerfällt dabei in eine Vielzahl selbständiger Kolonien sowie in stabile und blinkende Figuren. Bei einem Test dieser Figur über mehrere tausend Mikronen, bei dem ein IBM PC einige Stunden brütete, konnte kein definierter Endzustand gefunden werden. Problematisch ist bei derartigen Ausdehnungen natürlich immer der begrenzte Bildschirm des Computers.

Ein weiteres interessantes Fünf-Mik ist ein sogenannter Segler (Bild 3), der mit viertelter »Lichtgeschwindigkeit« schräg durch Mikronesien gleitet. Mit einem schnellen Programm erscheint der Segler wie ein Lebewesen, das schwanzwedelnd über die Bildfläche »wuselt«. Einen nahen Verwandten des Fünf-Mik nach Bewegung der Figur zeigt Bild 4. »Lichtgeschwindigkeit« in Mikronesien ist als höchstmögliche Geschwindigkeit die Verschiebung der Figur um ein Feld in jeder Mikrone. Was dem einzelnen Mik nicht erlaubt ist, realisiert er also im Kollektiv: kontinuierliche Bewegung. Daß die Dynamik dabei mit ständigem Opfern und Neugeborenwerden bezahlt wird, ist nicht die einzige Analogie von »Leben« zum wahren Leben.

Spielen Sie Schöpfer

Conway glaubte zunächst nicht, daß Figuren bis in alle Unendlichkeit weiterwachsen könnten. Er setzte deshalb in Martin Gardners Mathematik-Kolumne im »Scientific American« einen Geldpreis für den Beweis des Gegenteils aus. Dieser Preis war innerhalb eines Monats vergeben. Eine Gruppe junger Mathematiker um R. William Gosper jr. hatte eine Seglerkanone entdeckt, die die uns bereits bekannten Segler regelmäßig in Abständen von 30 Mikronen aussendet (Bild 5). Die Kanone selbst schwingt währenddessen hin und her, das heißt sie erreicht ihre alte Form jeweils wieder nach 30 Zeiteinheiten.

Ein anderer Amerikaner machte kurze Zeit darauf von sich reden. Er hatte eine Konstellation von 30 Seglern entdeckt, die beim Zusammenstoßen eine Seglerkanone bildeten. Nur wenig später erfand Gospers Mannschaft eine neue, eigenwillige Mik-Kolonie. Es handelt sich dabei um ein sogenanntes »Pentadekathlon« (Bild 6). Dieses Gebilde, das sich alle 15 Mikronen wiederholt, geht aus einer sehr einfachen Anfangsfigur hervor: Eine ununterbrochene waagerechte Reihe genügt zur Anregung des Pentadekathlon, die Anfangsfigur selbst wird später jedoch nicht mehr erreicht. Dieser Oszillator (so nennt man regelmäßig schwingende Objekte) kann noch einiges

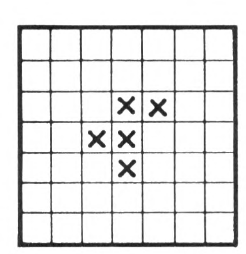


Bild 2. Das »r-Pentanino« dehnt sich als kleinste Figur unendlich weit aus

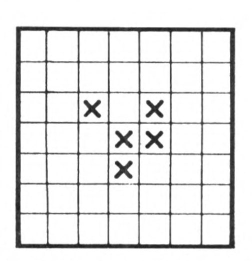


Bild 3. Der »Fünf-Mik-Segler« durchrast mit viertelter Lichtgeschwindigkeit das Spielfeld

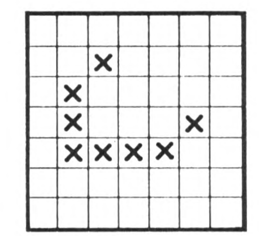


Bild 4. Der »Acht-Mik-Segler« hat die endgültige Lichtgeschwindigkeit erreicht

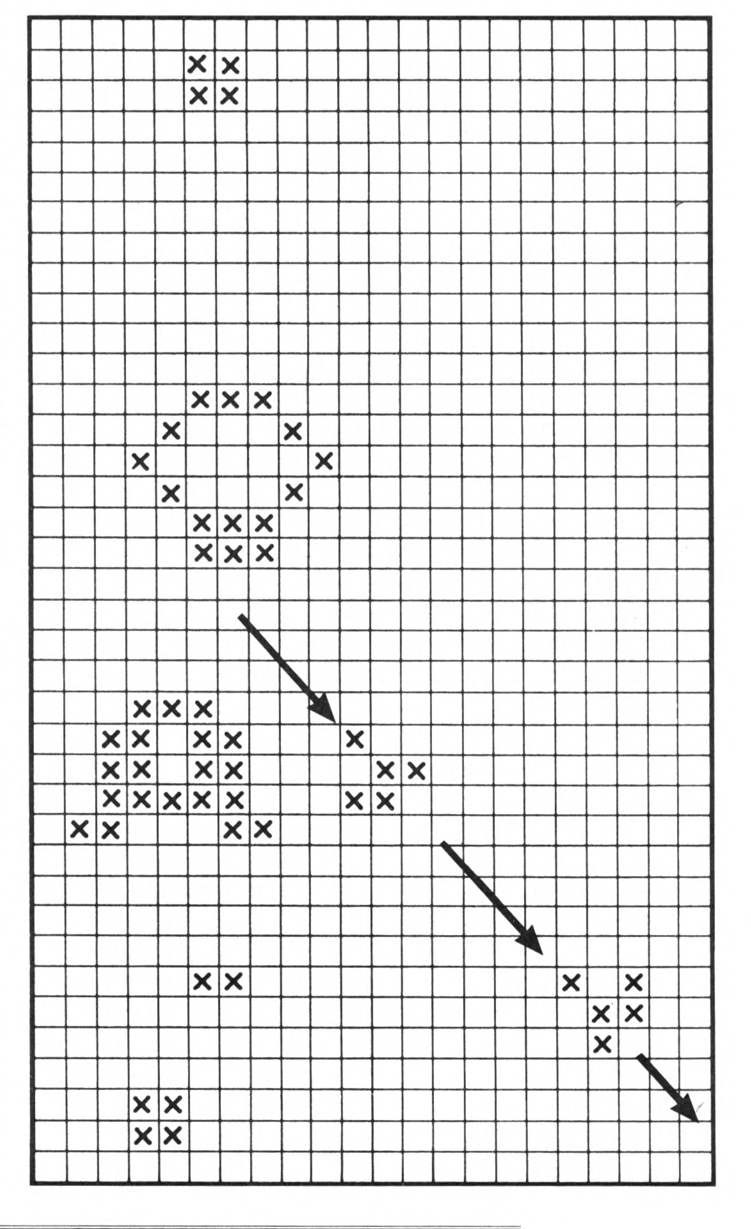


Bild 5. »Gospers Seglerkanone« regeneriert sich selbst. Sie erreicht den Urzustand nach 30 Zeiteinheiten.

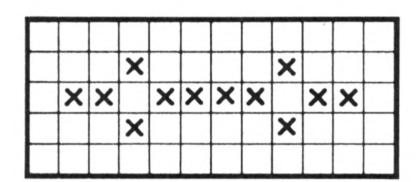


Bild 6. Das »Pentadekathlon« repräsentiert eine neue recht eigenwillige Mik-Kolonie

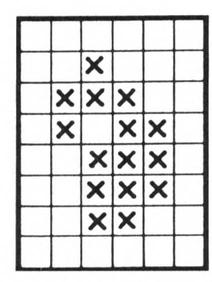


Bild 7. Der »15-Mik-Segler« erreicht immerhin beachtliche halbe Lichtgeschwindigkeit

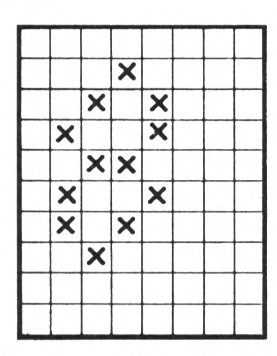


Bild 8. Aus der »schiefen Acht« bilden sich nach vier Mikronen zwei Segler

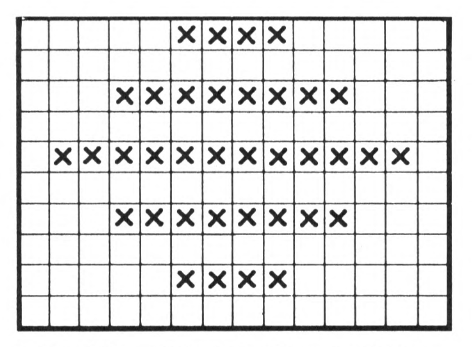


Bild 9. Die »4-8-12-Kolonie« entsendet ganze vier Segler in alle Richtungen

mehr. An der richtigen Stelle angeordnet, frißt er die Segler aus der Seglerkanone auf oder wirft sie zurück. Dementsprechend können zwei Pentadekathlons in richtiger Stellung einen Segler hin und her schießen. Man kann aber auch dafür sorgen, daß sich Seglerströme schneiden. Hierzu müssen mehrere Kanonen entsprechend plaziert und beim Aufbau um 90 oder 270 Grad gedreht werden. Man erhält so sehr ausgefallene Ergebnisse. Eine der interessantesten Entwicklungen auf diesem Gebiet ist eine Art »Werft«, die in zwei kreuzenden Seglerströmen errichtet wird. Diese Werft entsendet ihrerseits nach einigen hundert Stufen einen Segler der größeren »Bootsklasse«. Ein absolutes »Dickschiff« zeigt Bild 7. Dieses Modell erreicht trotz seiner Größe immerhin die beachtliche halbe Lichtgeschwindigkeit. Es bewegt sich in vier Mikronen um zwei Zellen abwärts.

Daneben existiert noch eine ganze Reihe einfacher Seglerkanonen. Diese entsenden einen oder mehrere Segler, um anschließend ihr Leben auszuhauchen. Die »schiefe Acht« (Bild 8) bildet nach vier Mikronen zwei Segler, die sich voneinander entfernen. Die »4-8-12-Kolonie« (Bild 9) schickt sogar vier Segler in alle Himmelsrichtungen davon. Sie braucht dazu 15 Mikronen.

Von diesen beiden Kanonen ist anschließend nichts übrig. Es gibt aber auch Kanonen, die ihre Segler verschießen und zudem deutliche Spuren oder ein heilloses Chaos zurücklassen. Daß sich mit Seglern auch weitaus sinnvollere Dinge vollbringen lassen als »schießen«, werden Sie später sehen.

Neben den Seglern und deren Fabriken existiert noch eine ganze Reihe reizvoller Anfangswelten. Die sogenannte »Erntemaschine« (Bild 10) zieht sich an einer beliebig langen Stange aufwärts. Zurück bleiben kompakte Viererblocks. Ein interessantes Experiment mit vielen Varianten ist das Impfen stabiler Welten mit »Viren«. Ein bekanntes Beispiel ist ein Gebilde aus Viererblocks (Bild 11), in dessen Mitte ein einziges Mik eingepflanzt wird. Dieser Störenfried erzeugt eine kreisförmige Explosion, die sich mit Lichtgeschwindigkeit konzentrisch ausbreitet. Wird der Virus jedoch um eine Zelle nach links gesetzt, so »repariert« sich die Figur innerhalb von zwei Mikronen.

Die Evolution der Ursuppe

Ein eigenes Kapitel könnte den schon angesprochenen Oszillatoren gewidmet werden. Um diese zu erzeugen, genügt oftmals eine symmetrische Anfangswelt. Symmetrien bleiben in jeder Population über die gesamte Lebenszeit erhalten, sofern keine Störung von außen eintritt. Einige Figuren bilden während ihres kurzen Lebens sehr hübsche Kaleidoskope, blähen sich dabei weit auf, um ebenso schnell wieder zu verschwinden oder in stabile und blinkende Häufchen zu zerfallen. Naturgemäß entwickeln sich Symmetrien weitaus häufiger in diese Richtung, als daß sie sich für den oszillierenden Endzustand entscheiden wie zum Beispiel das Pentadekathlon. Ein primitiver Oszillator wird von sechs Miks gebildet (Bild 12). Der »Kelch« (Bild 13) schwingt in die Spiegelverkehrte und wieder zurück. Wie sich das »Karo« (Bild 14) verhält, sollte jeder einmal selbst ausprobieren.

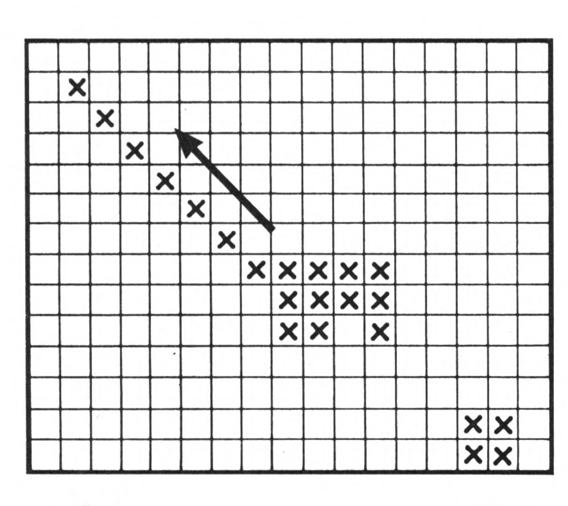


Bild 10. Die »Erntemaschine« hinterläßt einfach kompakte Viererblocks

Bild 11. Schon ein Mik kann das ansonsten stabile Gitter zerstören

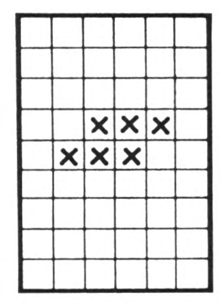


Bild 12. Der »Blinker« besteht aus sechs Zeilen, die sich nicht vermehren

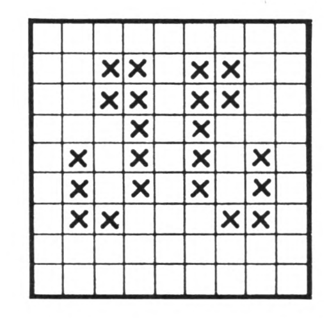


Bild 13. Der »Kelch« schwingt in die Spiegelverkehrte und wieder zurück

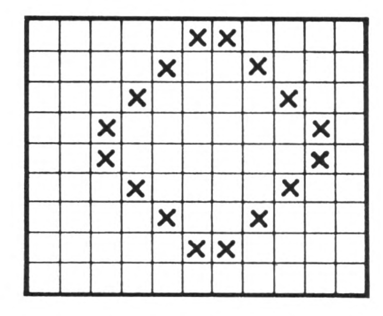


Bild 14. Das »Karo« besteht aus 16 einzelnen Zeilen, symmetrisch angeordnet

Sicherlich hat der eine oder andere sich schon gefragt, welchen Sinn dieses Spiel erfüllt. Amüsement für neurotische Mathematiker etwa – oder purer Selbstzweck? Daß das Spiel durchaus von einer philosophischen Seite aus betrachtet werden kann, aus der sich auch für andere Wissenschaften weitreichende Konsequenzen ergeben, wollen wir an dieser Stelle beleuchten.

Wir haben »Leben« bis hierher in der Grundform mit einigen recht einfachen Anfangsfiguren kennengelernt. In diesem Stadium »hinken« Vergleiche mit biologischen Vorgängen in der Natur sicherlich sehr stark. Jedoch bietet »Leben« eine hervorragende Basis, um auch weitaus komplexere Zusammenhänge zu simulieren. Hierzu können einerseits die Regeln erweitert werden, oder es müssen komplexere Anfangswelten aufgebaut werden, in der gewisse Eigenschaften verschiedener Kolonien gezielt eingesetzt werden. Derartige Experimente können allerdings nur mit komplizierten Vorausberechnungen realisiert werden, da das Herumprobieren mit größeren Kolonien nur selten zu befriedigenden Ergebnissen führt. Dazu reicht einerseits die uns zur Verfügung stehende Rechnerkapazität nicht aus, andererseits würden allein die mathematischen Theorien bereits genügen, um dieses Heft vollständig zu füllen. Nur ansatzweise sollen deshalb einige der Bedeutungen dieses aufregenden Spiels genannt werden.

Es lassen sich zum Beispiel auf der Basis von Leben zweidimensionale Computer konstruieren. Eine Seglerkanone entspricht dabei einem Taktgeber, die Segler selbst dienen gleichermaßen als elektrische Impulse oder Nachrichtenträger. Wie wir schon gesehen haben, lassen sich durch Zusammenstöße von Seglern die verschiedensten Zustände erreichen. Damit steht auch der Konstruktion logischer Gatter nichts mehr im Wege. Die einfachtse Funktion, der Inverter (Not-Gatter, Bild 15), besteht aus einer Seglerkanone, die eine kontinuierliche Impulsfolge

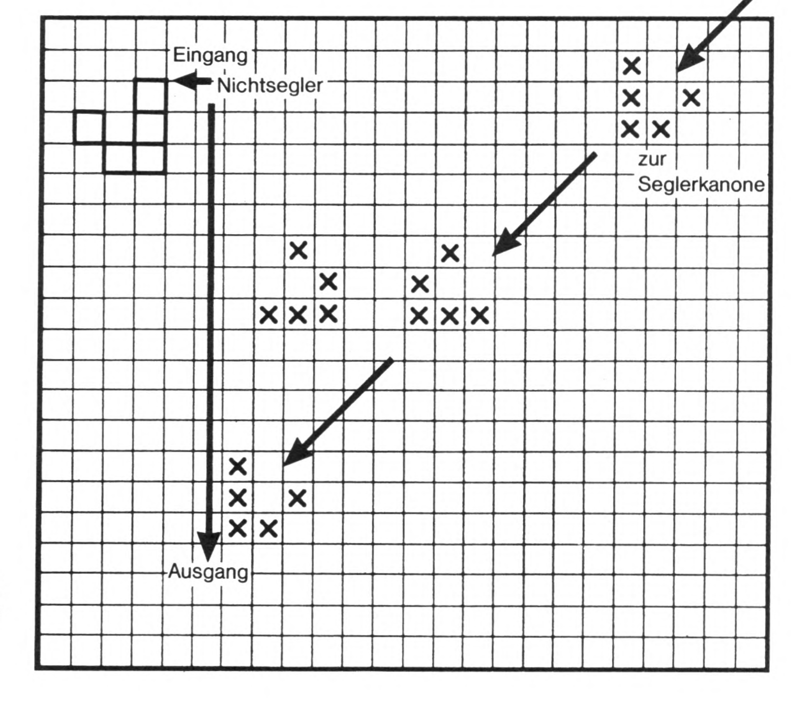


Bild 15. Der »Inverter«: Eine Seglerkanone sendet kontinuierliche Impulsfolgen aus

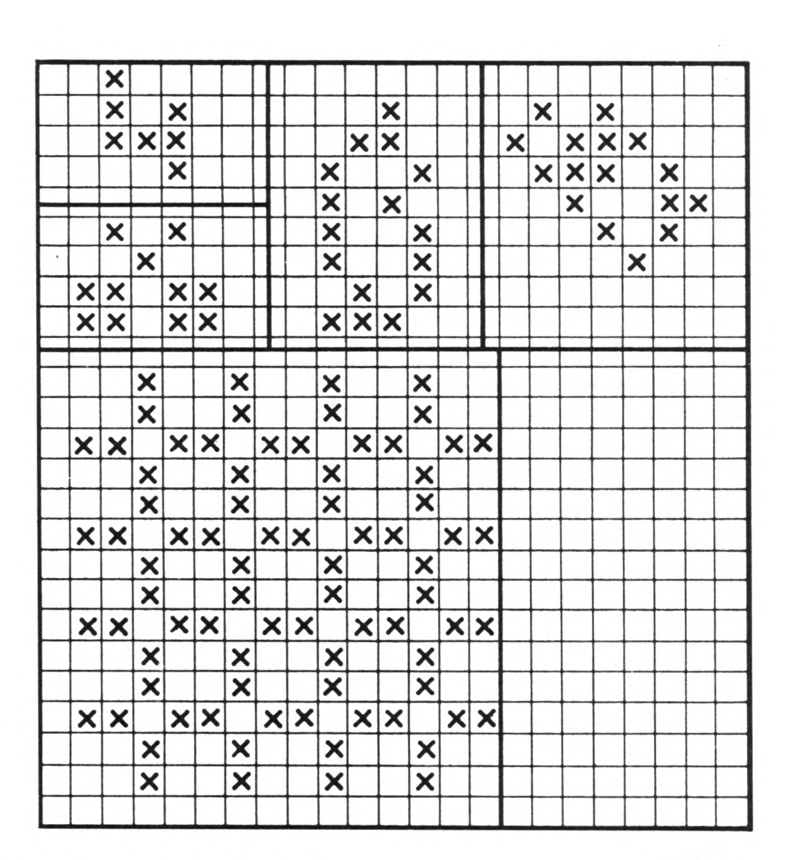


Bild 16. Diese vier Ausgangspopulationen erzeugen interessante Bilder

aussendet. Auf diesen Gleiterstrom wird senkrecht der Strom der Eingangssignale gesendet, in der Weise, daß kollidierende Segler sich gegenseitig auslöschen. Einer logischen 1 im Eingangssignal entspricht dabei ein vorhandener, einer logischen 0 ein nicht vorhandener Segler. Taucht nun ein »Nichtsegler« (0) am Kollisionsort auf, so kann der Segler des Kanonenstroms unbeschadet passieren, das bedeutet aus der 0 wird im Ausgang eine 1 und umgekehrt. Bemerkenswert ist, daß das eigentliche NOT-Gatter, der Kollisionsort, keine eigene Struktur benötigt.

Dementsprechend lassen sich ebenfalls AND-Gatter und OR-Gatter aufbauen und durch Verknüpfung mit dem NOT-Gatter selbstverständlich in NAND- und NOR-Funktionen umwandeln. Diese »Schaltungen« sind aber bereits derart kompliziert, daß sie auf einem Heimcomputer nicht programmiert werden können.

Ebenso schwierig, aber machbar, sind Register. Diese bestehen aus stabilen Viererblocks, die durch gezieltes Beschießen mit Seglern um drei Zellen vor- oder zurückgesetzt werden können. Logisch 0 oder 1 sind hier von der Plazierung abhängig. Berücksichtigt man die komplexen Strukturen, die einfache logische Funktionen bereits aufwerfen, so übersteigen die Ausmaße, die ein Allzweck-Life-Computer benötigt, die Grenzen der Vorstellung.

Wer sich über diese Art Computer näher informieren möchte, dem sei das vierbändige Werk »Gewinnen – Strategien für mathematische Spiele« empfohlen. Es wurde von John Horton Conway, Richard K. Guy und Elwyn R. Berlekamp verfaßt und ist im Vieweg-Verlag erschienen. »Leben« befindet sich in Band 4.

Eine ganz andere, aber kaum weniger wichtige Frage ist, ob Gebilde existieren, die in der Lage sind, sich selbst zu reproduzieren. Andere Mathematiker suchen nach dem genauen Gegenteil: Ist ein sogenannter »Garten Eden« möglich; eine Anfangswelt, die im Verlauf eines Spiels nicht entstehen kann? Ein solches »Paradies« könnte sich demzufolge nicht selbst reproduzieren und müßte, da es keine Vorläufer hat, vom Spieler selbst erschaffen werden. Glaubt man den Berechnungen, so ist hierzu ein Quadrat mit der Seitenlänge von einigen Milliarden Zellen erforderlich.

Das Programm, ein Reiz für sich

Betrachtet man dagegen die Theorie der »reproduzierenden Zellular-Automaten«, drängt sich der Vergleich mit der Entstehung des wirklichen Lebens auf. Laut einer Theorie entstanden die ersten primitiven Lebensformen dieser Erde aus einer Ursuppe von Aminosäuren und reproduzierten sich aus den diesen innewohnenden einfachen Strukturen, einer Art »Urerbinformation«. »Leben« könnte demnach ein mathematisches Modell von der Entstehung des Lebens sein. Conway sagte dazu einmal: »Irdisches Leben ist komplizierter, als es sein muß. Statt der 92 Elemente genügen eigentlich zwei: aus und ein!«. Ist vielleicht das ganze Universum ein gigantischer zellulärer Automat?

Wem eine Welt aus lauter Nullen und Einsen der Ketzerei gleichkommt, dem schlagen wir eine Erweiterung der Spielregeln vor. Diese Regeln lassen sich natürlich ebenso programmieren. In vielen Fällen müssen nur die Zustände, die eine tote oder lebendige Zelle annehmen kann, erhöht werden. Man kann auf diese Weise auf wesentlich kleineren Feldern die »Evolution« nachvollziehen:

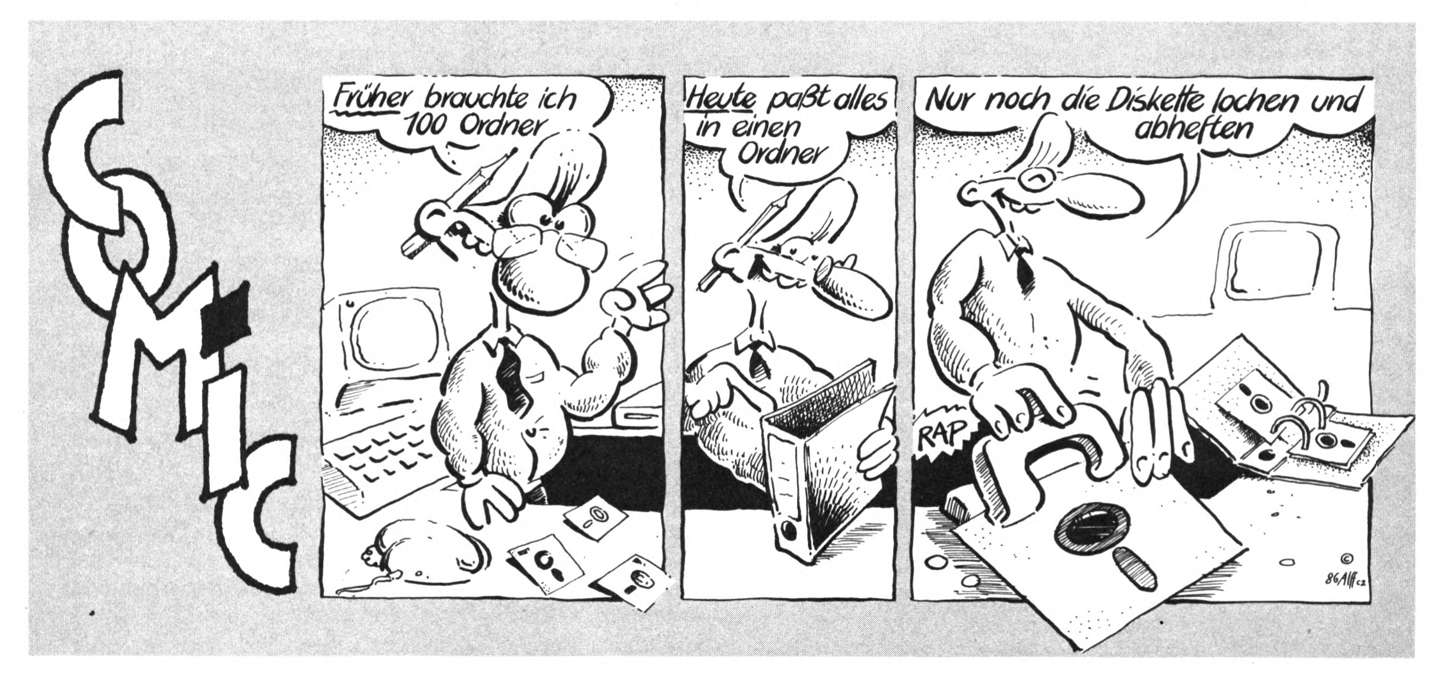
1. Programmieren Sie Freß- und Nahrungs-Miks. Es wäre zum Beispiel denkbar, daß Freß-Miks ohne Nahrung sterben, sich mit ihr aber schnell vermehren.
2. Bauen Sie Zonen unterschiedlicher Eigenschaften im Life-Universum mit ein. Denkbar wären »Schwarze Löcher«, in deren Umgebung jegliches Leben ausstirbt.
3. Geben Sie auserwählten Miks oder ganzen Kolonien Unsterblichkeit. In der unmittelbaren Nähe solcher Kolonien entsteht ständig neues Leben.
4. Möglich ist auch eine automatische Begrenzung der Lebenszeit. Überlegen Sie sich Regeln für Krankheit, Alter, Selektion, Tod.
5. Schaffen Sie Regeln für Wanderkolonien.

Sicherlich fallen Ihnen noch mehr Variationen ein, um die erweiterten Regeln »lebensnah« zu gestalten. Der Phantasie und dem Programmierspaß sind keine Grenzen gesetzt.

Wer brennt nach so viel Theorie nicht darauf, endlich etwas auf seinem Monitor zu sehen? Doch bevor Sie sich auf das Abtippen des Programms stürzen (falls nicht schon längst geschehen), empfehlen wir noch etwas Geduld. Es lohnt sich allemal, erst etwas Denkarbeit in die Suche nach einem effektiven Algorithmus zu investieren.

Das hier vorgestellte Programm realisiert vereinfacht den Algorithmus, den Sie im folgenden vorfinden. Es ist leicht überschaubar und sehr einfach zu erweitern.

Doch nun zum Ablaufschema. Zunächst einmal müssen wir einen Editor



für unsere Simulation anlegen. Dazu müssen die Ausmaße unseres Spielfeldes bestimmt werden. Es bietet sich der Zeichenbildschrim an, der in unserem Programm benutzt wird. Natürlich wäre es reizvoller, auf hochauflösende Grafik zurückzugreifen. Jedoch erhöht sich damit die Zahl der Berechnungen extrem. Was das unter Basic bedeutet, weiß jeder, der schon mal programmiert hat. Das Programm wird sehr langsam.

Im Modus 1 stehen uns immerhin 38 mal 23 Zellen zur Verfügung (40x25 minus Rand), was für die meisten der gezeigten Figuren ausreicht. Nach dem Start des Programms steht der Cursor in der Mitte des Bildschirms und läßt sich mit den Cursortasten auf die gewünschten Positionen bringen. Mit der Copy-Taste können Miks gesetzt und wieder gelöscht werden. Der Zeichensatz unseres CPC kommt der Symbolik unseres Spiels sehr entgegen. Wir haben als Miks die Männchen (CHR$(249)) gewählt. Wem andere Zeichen übersichtlicher erscheinen, setzt nur einen anderen Wert ein (beispielsweise CHR$(143)). Sie müssen aber alle CHR$(249) ersetzen! Ist die Anfangspopulation bestimmt, wird mit der Leertaste gestartet. Mit »A« kann das Programm jederzeit abgebrochen werden.

Der Editor (Zeile 120 bis 280) besteht im wesentlichen aus einer Tastenabfrage. Hier werden auch die gesetzten Zellen in einem zweidimensionalen Stringfeld »(b$(s,z)« (siehe auch Tabelle) abgelegt. Da nur in der Umgebung belebter Zellen ein Ereignis stattfinden kann, werden zur Einsparung von Rechenzeit die Variablen minx, miny, maxx und maxy angelegt. Diese Variablen dienen der Kennzeichnung der linken oberen und rechten unteren Ecke eines »Rechenfensters«. Es handelt sich hierbei natürlich um kein Window, sondern es wird lediglich der Bereich festgelegt, der später vom Programm überprüft wird. Diese Variablen müssen während des Editierens und nach der Berechnung der neuen Positionen ständig aktualisiert werden (Zeilen 230 bis 570).

Variable	Bedeutung
b$ =	Bildschirmfeld
minx, maxx =	Begrenzung des »Rechenfensters« in x-Richtung
miny, maxy =	Begrenzung des »Rechenfensters« in y-Richtung
s =	aktuelle Cursorposition (Spalte)
z =	aktuelle Cursorposition (Zeile)
x, y =	Spalte, Zeile der Umgebung der Zelle (s,z)
l =	Anzahl der belebten Nachbarzellen
ugx, ogx =	neue Begrenzung des »Rechenfensters« (x-Richtung)
ugy, ogy =	neue Begrenzung des »Rechenfensters« (y-Richtung)

Tabelle. Die Variablenliste zu »Leben«

```
10 '************************************
   *******                              [E66C]
20 '*****          L I F E    1.2
    *****                               [B87C]
30 '*****==============================
   ==*****                              [1456]
40 '*****Copyright 1986 by Th. Stopfkuch
   en*****                              [255E]
50 '************************************
   *******                              [AE74]
60 CLEAR:MODE 1:DIM b$(41,28):BORDER 6:F
   OR i=0 TO 41:LOCATE 1,1:PRINT i:FOR j
   =0 TO 28:b$(i,j)=" ":NEXT j:NEXT i:CL
   S                                    [6AEE]
70 INK 3,24:FOR i=1 TO 40:LOCATE i,1:PRI
   NT CHR$(143);:LOCATE i,25:PRINT CHR$(
   143);:NEXT:FOR i=2 TO 24:LOCATE 1,i:P
   RINT CHR$(143);:LOCATE 40,i:PRINT CHR
   $(143);:NEXT                         [95F4]
80 INK 2,0:PEN 2:PAPER 3                [A718]
90 LOCATE 5,1:PRINT"L I F E - Erleben Si
   e das Leben";                        [E104]
100 PEN 1:PAPER 0:LOCATE 20,12:CALL &BB8
    A                                   [BF06]
110 minx=41:maxx=0:miny=26:maxy=0       [50EE]
120 '--- Eingabe der Anfangskonfiguratio
    n ---                               [D318]
130 IF INKEY$="" THEN 130               [6D16]
140 IF INKEY(0)<>-1 THEN GOTO 250       [0038]
150 IF INKEY(1)<>-1 THEN GOTO 260       [253E]
160 IF INKEY(2)<>-1 THEN GOTO 270       [5E44]
170 IF INKEY(8)<>-1 THEN GOTO 280       [4A54]
180 IF INKEY(47)<>-1 THEN 300           [9FFC]
190 IF INKEY(9)=-1 THEN 130             [A322]
200 s=POS(#0):z=VPOS(#0)                [F7AA]
210 IF b$(s,z)=CHR$(249) THEN b$(s,z)="
    " ELSE b$(s,z)=CHR$(249)            [BF0E]
220 CALL &BB8D:PRINT b$(s,z):LOCATE s,z:
    CALL &BB8D                          [1068]
230 minx=MIN(s,minx):miny=MIN(z,miny):ma
    xx=MAX(maxx,s):maxy=MAX(maxy,z)     [7B4A]
240 GOTO 130                            [F046]
250 IF VPOS(#0)<3 THEN 130 ELSE CALL &BB
    8D:PRINT CHR$(11);:CALL &BB8A:GOTO 1
    30                                  [754A]
260 IF POS(#0)>38 THEN 130 ELSE CALL &BB
    8D:PRINT CHR$(9);:CALL &BB8A:GOTO 13
    0                                   [3DC2]
270 IF VPOS(#0)>23 THEN 130 ELSE CALL &B
    B8D:PRINT CHR$(10);:CALL &BB8A:GOTO
    130                                 [45B4]
280 IF POS(#0)<3 THEN 130 ELSE CALL &BB8
    D:PRINT CHR$(8);:CALL &BB8A:GOTO 130
                                        [4550]
290 '--- Berechnung der neuen Positionen
     ---                                [241E]
300 CALL &BB8D:EVERY 10 GOSUB 690:LOCATE
     10,25:PAPER 3:PEN 2:PRINT "Generati
    on Nr. :{2 SPACE}"                  [0926]
310 g=g+1:LOCATE 28,25:PAPER 3:PEN 2:PRI
    NT g:PAPER 0:PEN 1                  [4FB8]
320 FOR z=miny-1 TO maxy+1              [2D18]
330 IF z<2 OR z>24 THEN GOTO 480        [E360]
340 FOR s=minx-1 TO maxx+1              [000A]
350 IF s<2 OR s>39 THEN GOTO 470        [9252]
360 LOCATE s,z:PRINT CHR$(22)+CHR$(1):LO
    CATE s,z:PRINT CHR$(144)            [BDE4]
370 IF b$(s,z)=" " THEN l=0 ELSE l=-1   [0664]
380 FOR y=z-1 TO z+1                    [5F12]
390 FOR x=s-1 TO s+1                    [AAF6]
400 IF b$(x,y)=CHR$(249) OR b$(x,y)=CHR$
    (43) THEN l=l+1                     [4860]
410 NEXT x                              [0D18]
420 NEXT y                              [A11C]
430 LOCATE s,z:PRINT CHR$(22)+CHR$(0):LO
    CATE s,z:PRINT b$(s,z):             [385C]
440 IF l=2 AND b$(s,z)=CHR$(249) OR l=3
    AND b$(s,z)=CHR$(249) THEN 470      [6CD6]
450 IF l=3 AND b$(s.z)=" " THEN b$(s.z)=
    CHR$(42):LOCATE s,z:PRINT CHR$(42); [7482]
460 IF b$(s,z)=CHR$(249) THEN b$(s,z)=CH
    R$(43):LOCATE s,z:PRINT CHR$(43);   [E3C4]
470 NEXT s                              [331A]
480 NEXT z                              [DF2A]
490 '--- Setzen der berechneten Zeichen
    ---                                 [9F1E]
500 ugx=41:ogx=0:ugy=26:ogy=0           [7A94]
510 FOR z=miny-1 TO maxy+1              [0B1A]
520 IF z<2 OR z>24 THEN GOTO 580        [B164]
530 FOR s=minx-1 TO maxx+1              [D80C]
540 IF s<2 OR z>39 THEN GOTO 570        [D964]
550 IF b$(s,z)=CHR$(42) THEN b$(s,z)=CHR
    $(249):LOCATE s,z:PRINT CHR$(249):GO
    SUB 660                             [D6A8]
560 IF b$(s,z)=CHR$(43) THEN b$(s,z)=" "
    :LOCATE s,z:PRINT " ":GOSUB 660     [8778]
570 NEXT s                              [1C1C]
580 NEXT z                              [C02C]
590 minx=ugx:maxx=ogx:miny=ugy:maxy=ogy [F238]
600 IF minx>maxx THEN 320               [8E42]
610 IF minx<2 OR miny<2 THEN minx=2:miny
    =2                                  [4A38]
620 IF maxx>39 THEN maxx=39             [884A]
630 IF maxy>24 THEN maxy=24             [4638]
640 GOTO 310                            [164E]
650 '--- Bestimmung der neuen Berechnung
    sgrenzen ---                        [24DC]
660 ugx=MIN(s,ugx):ogx=MAX(s,ogx)       [1ABC]
670 ugy=MIN(z,ugy):ogy=MAX(z,ogy)       [CEE2]
680 RETURN                              [AB3C]
690 in$=INKEY$:in$=LOWER$(in$):IF in$="a
    " THEN GOTO 60 ELSE RETURN          [8B86]
700 LOCATE 40,1:PRINT"a"                [9178]
710 GOTO 710                            [F952]
```

Listing. »Leben« ist unter Basic leicht zu programmieren

Nach dem Ende der Eingabe und der Zuordnung der entsprechenden Variablen beginnt das Programm mit der eigentlichen Bearbeitung (Zeile 290 bis 640).

Innerhalb des »Rechenfensters« werden alle Zellen auf die Spielregeln hin überprüft. Für jede Zelle muß der Inhalt der acht Nachbarfelder gelesen werden (Zeile 380 bis 420). Dabei ist die Variable »I« Zähler der belebten Nachbarfelder. Die Zeilen 430 bis 450 bilden den Kern des Programms. Die Regeln sind hier in nur drei Zeilen codiert. Als Besonderheit arbeitet das Programm mit zwei weiteren Symbolen: Ein Kreuz (CHR$(43)) zeigt an, daß der Mik im folgenden Zeittakt sterben wird, ein Sternchen (CHR$(42)) bedeutet, daß in einer unbelebten Zelle ein Mik geboren wird. Nach der Überprüfung werden die zusätzlichen Symbole angepaßt. So wird ein Kreuz gelöscht, ein Sternchen wird in einen Mik umgewandelt (Zeile 490 bis 640). An dieser Stelle wird ebenfalls das »Rechenfenster« aktualisiert. In der letzten Zeile fragt das Programm alle 0,2 Sekunden, ob eine Taste gedrückt wurde.

Haben Sie das Programm verstanden und sich nicht durch die vierfach verschachtelten FOR-NEXT-Schleifen abschrecken lassen? Dann werden Sie zum Schluß sicher die Tips interessieren, die das Programm beschleunigen.

Viele Wege führen nach Rom. Doppelt so viele führen vom Problem zum Programm. Weil das ganz besonders für unser Programm gilt, können wir hier nur einige Tips geben, »Leben« effektiver zu programmieren.

Besser leben

Das erste Ziel wird natürlich sein, die hochauflösende Grafik zu benutzen. Bei einer Auflösung von 320 x 200 Punkten müssen 64000 Bildpunkte berücksichtigt werden. Nun würde aber bei üblicher Array-Speicherung schon das Feld allein unsere Speicherkapazität weit überschreiten. Man benutzt daher folgenden Trick. Für jede Zelle gelten genau vier Zustände: Zelle tot, Zelle lebendig, tote Zelle wird bei t+1 lebendig, lebende Zelle stirbt bei t+1. Man belegt daher für jede Zelle nur noch Halbnibbles (2 Bit). So schrumpft der Speicherbedarf für eine 320x200-Spielwiese auf 16 KByte. Der Nachteil dieser Methode ist, daß die Zeilen- und Spaltenvariablen nicht mehr gleichzeitig als Feldkoordinaten benutzt werden können. Das Beschreiben und Abfragen der Halbnibbles muß mit AND und OR geschehen.

Bei unserem Programm wurde der Rechenaufwand durch Bestimmen eines »Rechenfensters« gemindert. So werden jedoch immer viele nicht relevante Zellen abgefragt. Man legt daher bei einem »hochauflösenden« Programm ein weiteres Feld an, in dem die Koordinaten der lebendigen Zellen und deren Nachbarzellen gespeichert werden. Ferner sorgt man dafür, daß keine Doppeleintragungen stattfinden. Auf diese Weise wird keine Abfrage zweimal vorgenommen. Statt daß, wie in unserem Programm, das Fenster gedehnt wird, wandern jetzt die Bereiche mit den Figuren über das Spielfeld.

Will man die Spielregel für »Life« erweitern, muß ein dreidimensionales Feld angelegt werden. Die »Lebensinformationen« stehen dann in der dritten Ebene.

»Leben« ist nicht nur ein interessantes Spiel, es bietet auch ein grenzenloses Betätigungsfeld, um Programmiertechniken zu üben und zu erlernen. Bild 16 zeigt noch andere bemerkenswerte Testkombinationen. So wird das Problem beispielsweise gern an Informatikkursen der Hochschulen gestellt. Zu guter letzt noch eine Warnung: So mancher Mathematiker soll von »Leben« schon süchtig geworden sein ...

(Matthias Rosin/hg)



Nur Fliegen ist schöner



Wer würde nicht gern einmal eine Boeing 727 als Pilot fliegen? Mit »Jetliner« können auch Sie auf allen größeren Verkehrs- und Militärflughäfen Deutschlands starten und landen.

Der Simulator hat die Daten von 43 Navigationsfunkfeuern, 20 Verkehrs- und neun Militärflugplätzen gespeichert. Sie können während des Fluges auf jedes Funkfeuer einstellen und entsprechend navigieren. Bei der Landung sind die Ost-West-Lage der Landebahnen, ihre individuellen Höhen und unterschiedlichen Längen besonders zu beachten, da Sie innerhalb der Landebahngrenzen starten, wieder aufsetzen und ausrollen müssen.

Da es sich um einen reinen Instrumentenflug handelt, stehen folgende Anzeigen zur Verfügung:
- HGS: Geschwindigkeit in Knoten (1 Knoten = 1,9 km/h) Warnlampe und -ton, wenn HGS über 500 Knoten steigt.
- VGS: Vertikalgeschwindigkeit in Fuß (1 Fuß = 32 cm). Warnlampe und -ton, wenn VGS unter minus 8000 Fuß fällt.
- HÖH: Höhe des Flugzeugs in Fuß. Wenn der Abstand zwischen Flugzeug und Boden kleiner als 300 Fuß ist, wird die relative Höhe zwischen Boden und Flugzeug angegeben. In diesem Fall wird in der Anzeige ein »H« sichtbar, das zu blinken beginnt, wenn Sie beim Sinken dem Boden gefährlich nahe kommen, ohne daß das Fahrwerk ausgefahren wurde.

Sonst wird die absolute Höhe über Null angegeben.
- KOM: Der Kompaß, Anzeige in Grad.
- ALF: In diesem Fenster erscheint die Kennung des Funkfeuers, darunter die Richtung in Grad.
- ENT: Entfernung zwischen Flugzeug und Funkfeuer in Meilen (1 Meile = 1,6 km).
- HRZ: Der künstliche Horizont gibt die Lage des Flugzeugs an.
- LOB: Diese Anzeige tritt nur bei Boden- und Landebahnnähe in Erscheinung. Sie zeigt die Abstände zu den vier Landebahnseiten (V für vorne, H für hinten, L für links und R für rechts) in tausendstel Meilen an. Drehen Sie das Flugzeug um 180 Grad, so tauschen gegenüberliegende Anzeigen die Plätze.

Die Fahrwerkpositionen:
- = Fahrwerk unten
- = Fahrwerk oben
- = Fahrwerk klemmt; In diesem Fall müssen Sie das Fahrwerk wieder einfahren.
- K: Schubkraft (0 bis 9)
- MOT: Zustand der drei Motoren:

nicht ausgefüllt (rot) = Motor aus
dunkel ausgefüllt (rot) = Motor läuft an
dunkelgrün ausgefüllt (orange) = Motor auf halber Drehzahl (auch Motorausfall)
hellgrün ausgefüllt (gelb) = Motor auf voller Drehzahl
- TR: Der Treibstoffvorrat in Litern. Eine Warnlampe spricht an, wenn der Treibstoffvorrat unter 3000 Liter sinkt.

Maximalgeschwindigkeit

- mit Landeklappe 1	230 Knoten
- mit Landeklappe 2	210 Knoten
- mit Landeklappe 3	190 Knoten
- mit Landeklappe 4	170 Knoten
- wenn Fahrwerk unten	270 Knoten
- am Boden	190 Knoten
- bei Kurven am Boden	60 Knoten
- in der Luft	520 Knoten
Maximale Vertikalgeschwindigkeit	minus 9900 Fuß
beim Aufsetzen	minus 500 Fuß

Tastaturbelegung:
»M« = Hochziehen/Abheben
»I« = Runterdrücken/Aufsetzen
»J« = Linkskurve
»K« = Rechtskurve
»Z« = Anwählen eines neuen Funkfeuers (oder Joystick 0)

Nach Anforderung eines neuen Funkfeuers wird die Kennung gelöscht und Sie müssen eine neue eingeben (immer dreistellig, also beispielsweise: »D«, »M«, »SPACE«). Falls das eingestellte Funkfeuer zu einem Flugplatz gehört, wird die Bodenhöhe der Flugplatzhöhe angeglichen. Erscheint aber die alte Kennung wieder im Fenster, so liegt die Höhe des angewählten Flugplatzes über der augenblicklichen Flughöhe; Sie müssen dann noch höher steigen.
»Q« = Anhalten der Simulation
»W« = Wieder fortfahren
»A« = Fahrwerk einziehen
»S« = Fahrwerk ausfahren
»[« = Landeklappen ausfahren
»]« = Landeklappen einfahren
»0« bis »9« = Schubkraft:
»0« = Gegenschub (Bremsen, angezeigt als »B«)
»1« = Motoren anlassen, im Stillstand laufen lassen
»2« bis »8« = Manövrier- und Fluggeschwindigkeiten
»9« = Maximale Schubkraft zum Abheben; sehr hoher Treibstoffverbrauch!
Nach dem Landen:
»T« = Auftanken, Motor reparieren lassen
»E« = Ende des Spiels

Start- und Landebahnen werden grafisch auf dem Bildschirm dargestellt.

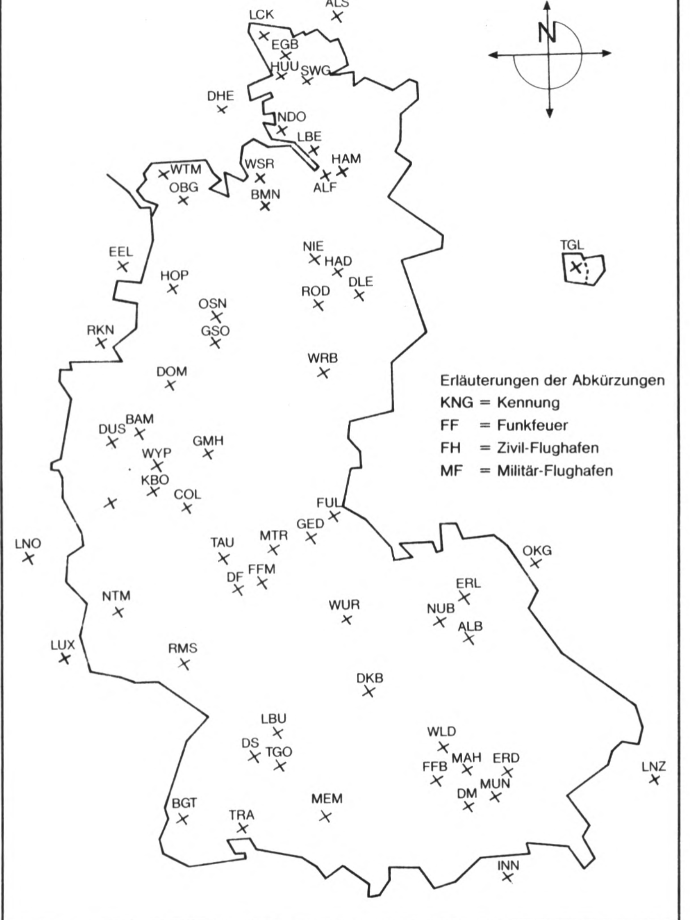
Ein Flug von München nach Nürnberg

Nun zur Sache: Sie wählen durch Drücken der Taste »4« München als Startflugplatz. Danach wird das Cockpit aufgebaut und die Anzeigen erscheinen. Jetzt lassen Sie mit der Taste »1« die Triebwerke an, erhöhen den Schub (»2«) und kurven so, daß der Kompaß 90 Grad zeigt, um parallel zur Landebahn zu stehen. Dann geben Sie maximalen Schub (»9«) und fahren bei einer Geschwindigkeit von zirka 90 Knoten die Landeklappen auf Stufe 2 aus (zweimal »[«). Bei einer Geschwindigkeit von zirka 120 Knoten heben Sie ab und ziehen das Fahrwerk ein (»A«). Wenn das Flugzeug schnell genug ist (ab zirka 180 Knoten), fahren Sie beide Landeklappen ein (zweimal »]«). Nun steigen Sie auf etwa 5500 Fuß und stellen auf das Funkfeuer ALB ein (»Z«, »A«, »L«, »B«), reduzieren den Schub auf Reisegeschwindigkeit (»7«) und fliegen darauf zu.

Bei einem Triebwerkausfall ist zu beachten, daß die Geschwindigkeit zurückgeht; den defekten Motor lassen Sie nach der Landung reparieren (»T«). Wenn ALB nur noch zirka 15 Meilen entfernt ist, stellen Sie den Flughafen Nürnberg (NUB) als Kennung ein und navigieren so, daß sich die Funkfeuerrichtung auf 270 Grad bewegt. Dabei reduzieren Sie den Schub (»5«) und gehen auf 2000 F hinunter. Wenn die

Peilung 270 Grad beträgt, schwenken Sie auf diese Richtung und gehen mit Schub 3 auf zirka 200 Fuß über Flughafenhöhe hinunter. Bei einer Geschwindigkeit unter 270 Knoten fahren Sie das Fahrwerk aus. Achten Sie bei der Landung besonders darauf, daß das Flugzeug innerhalb der Landebahngrenzen aufsetzt. Wenn Sie gelandet sind, fahren Sie alle vier Landeklappen aus und gehen auf Gegenschub (»0«). Nach dem Stillstand zeigt ein Druck auf die Taste »E« Flugzeit und Treibstoffverbrauch an. (Claus Herwig/ja)

KNG	FF	FH	MF	NAME	KNG	FF	FH	MF	NAME
ALB	x			Allersberg	LBE	x			Elbe
ALF		x		Hamburg/Fh	LBU	x			Luburg
ALS	x			Alsie	LCK			x	Leck
BAM	x			Barmen	LNO	x			Olno
BGT			x	Bremgarten	LNZ	x			Linz
BMN		x		Bremen	LUX	x			Luxemburg
COL	x			Cola	MAH	x			Maisach
DF		x		Frankfurt/Fh	MEM			x	Memmingen
DHE	x			Helgoland	MTR	x			Metro
DKB	x			Dinkelsbühl	MUN	x			München
DLE	x			Leine	NDO	x			Nordholz
DM		x		München/Fh	NIE	x			Nienburg
DOM	x			Dortmund	NTM	x			Nattenheim
DS		x		Stuttgart	NUB		x		Nürnberg
DUS		x		Düsseldorf	OBG	x			Oldenburg
EEL	x			Eelde	OKG	x			Cheb (Eger)
EGB			x	Eggebek	OSN	x			Osnabrück
ERD	x			Erding	RKN	x			Rekken
ERL	x			Erlangen	RMS			x	Remstein
FFB			x	Fürstenfeldbruck	ROD	x			Rodenberg
FFM	x			Frankfurt	SWG			x	Schleswig
FUL	x			Fulda	TAU	x			Taunus
GED	x			Gedern	TGL		x		Berlin/Tegel
GMH	x			Germinghausen	TGO	x			Tango
GSO	x			Gütersloh	TRA	x			Trasadingen
HAD		x		Hannover	WLD	x			Walda
HAM	x			Hamburg	WRB	x			Warburg
HOP			x	Hopsten	WSR	x			Weser
HUU			x	Husum	WTM	x			Wittmund
INN	x			Innsbruck	WUR	x			Würzburg
KBO		x		Köln/Bonn	WYP	x			Wipper



◀ Verzeichnis und Lageplan der enthaltenen Flughäfen

```
10 REM /\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\  [116A]
20 REM \/          JETLINER          \/  [21F2]
30 REM /\  For CPC by Claus Herwig   /\  [06E6]
40 REM \/    (c) January 1986        \/  [DA88]
50 REM /\                            /\  [6B3E]
60 REM \/  Orginal version by        \/  [8D9A]
70 REM /\   Roman Grandis (CP 2/85)  /\  [A530]
80 REM \/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/  [3F78]
100 RANDOMIZE TIME                       [A252]
110 GOSUB 6200                           [BF34]
120 MODE 1:PEN 1                         [FF2E]
130 DIM z$(25)                           [8108]
140 INK 0,1:INK 1,24:INK 2,15:INK 3,6    [2E28]
150 a4=0:lg=4                            [2F6C]
200 wvgs=4:svgs=21                       [8F0E]
210 wkr=38:skr=19                        [35C0]
220 wla1=7:wla2=14:sla=1                 [27AE]
230 wtr=34:str=23                        [29D6]
240 wmot=36:smot=21                      [6880]
250 wkom=11:skom=18                      [3C5C]
260 wkkom=11:skkom=21                    [12FE]
270 wkent=11:skent=23                    [DA04]
280 wwl1=3:swl1=21:wwl2=3:swl2=18:wwl3=3
    3:swl3=23:                           [8B7A]
290 whoe=3:shoe=23                       [9CD2]
300 wkl=36:skl=17:wkln=36:skln=18        [AF08]
310 wfw=33:sfw=17                        [D8B4]
320 wgs=4:sgs=18                         [9648]
330 wlav=26:slav=18:wlah=26:slah=22:wlal
    =24:slal=20:wlar=29:slar=20          [3BDC]
340 wff=13:sff=20                        [5D66]
400 lb$(1)="{CTRL J}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I}{CTRL I}
    {CTRL I}{CTRL I}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I}.{CTRL J
    }{CTRL H}{CTRL H} {CTRL J}{CTRL H}{C
    TRL H}/{CTRL J}{CTRL H}{CTRL H}/{CTR
    L J}{CTRL H}{CTRL H}/{CTRL J}{CTRL H
    }{CTRL H} {CTRL J}{CTRL H}{CTRL H}.{
    CTRL J}{CTRL H}{CTRL H} {CTRL J}{CTR
    L H}{CTRL H}/{CTRL J}{CTRL H}{CTRL H
    }/{CTRL J}{CTRL H}{CTRL H}/{CTRL J}{
    CTRL H}{CTRL H} {CTRL J}{CTRL H}{CTR
    L H}.{CTRL K}{CTRL K}{CTRL K}{CTRL K
    }{CTRL K}{CTRL K}{CTRL K}{CTRL K}{CT
    RL K}{CTRL K}{CTRL K}{CTRL K}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I}.{CTRL J} {CTRL J}\{CTRL J}
    \{CTRL J}\{CTRL J} {CTRL J}.{CTRL J}
     {CTRL J}\{CTRL J}\{CTRL J}\{CTRL J}
     {CTRL J}.{CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL K}{CTRL K}{CTRL K}{CTRL
    K}{CTRL K}{CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}"  [5B3E]
410 lb$(2)="{CTRL J}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I}{CTRL I}
    {CTRL I}{CTRL I}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I}/{CTRL J
    }{CTRL H}{CTRL H} {CTRL J}{CTRL H}{C
    TRL H}.{CTRL J}{CTRL H}{CTRL H} {CTR
    L J}{CTRL H}{CTRL H}/{CTRL J}{CTRL H
    }{CTRL H}/{CTRL J}{CTRL H}{CTRL H}/{
    CTRL J}{CTRL H}{CTRL H} {CTRL J}{CTR
    L H}{CTRL H}.{CTRL J}{CTRL H}{CTRL H
    } {CTRL J}{CTRL H}{CTRL H}/{CTRL J}{
    CTRL H}{CTRL H}/{CTRL J}{CTRL H}{CTR
    L H}/{CTRL K}{CTRL K}{CTRL K}{CTRL K
    }{CTRL K}{CTRL K}{CTRL K}{CTRL K}{CT
    RL K}{CTRL K}{CTRL K}{CTRL K}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I}\{CTRL J} {CTRL J}.{CTRL J}
     {CTRL J}\{CTRL J}\{CTRL J}\{CTRL J}
     {CTRL J}.{CTRL J} {CTRL J}\{CTRL J}
    \{CTRL J}\{CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL K}{CTRL K}{CTRL K}{CTRL
    K}{CTRL K}{CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
```

Listing. Fliegen ohne Pilotenschein



```
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}"  [60A0]
420 lb$(3)="{CTRL J}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I}{CTRL I}
    {CTRL I}{CTRL I}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I}/{CTRL J
    }{CTRL H}{CTRL H}/{CTRL J}{CTRL H}{C
    TRL H}/{CTRL J}{CTRL H}{CTRL H} {CTR
    L J}{CTRL H}{CTRL H}.{CTRL J}{CTRL H
    }{CTRL H} {CTRL J}{CTRL H}{CTRL H}/{
    CTRL J}{CTRL H}{CTRL H}/{CTRL J}{CTR
    L H}{CTRL H}/{CTRL J}{CTRL H}{CTRL H
    } {CTRL J}{CTRL H}{CTRL H}.{CTRL J}{
    CTRL H}{CTRL H} {CTRL J}{CTRL H}{CTR
    L H}/{CTRL K}{CTRL K}{CTRL K}{CTRL K
    }{CTRL K}{CTRL K}{CTRL K}{CTRL K}{CT
    RL K}{CTRL K}{CTRL K}{CTRL K}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I}\{CTRL J}\{CTRL J}\{CTRL J}
     {CTRL J}.{CTRL J} {CTRL J}\{CTRL J}
    \{CTRL J}\{CTRL J} {CTRL J}.{CTRL J}
     {CTRL J}\{CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL K}{CTRL K}{CTRL K}{CTRL
    K}{CTRL K}{CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}"  [47A4]
430 lb$(4)="{CTRL J}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I}{CTRL I}
    {CTRL I}{CTRL I}{CTRL I}{CTRL I}{CTR
    L I}{CTRL I}{CTRL I}{CTRL I} {CTRL J
    }{CTRL H}{CTRL H} {CTRL J}{CTRL H}{C
    TRL H} {CTRL J}{CTRL H}{CTRL H} {CTR
    L J}{CTRL H}{CTRL H} {CTRL J}{CTRL H
    }{CTRL H} {CTRL J}{CTRL H}{CTRL H} {
    CTRL J}{CTRL H}{CTRL H} {CTRL J}{CTR
    L H}{CTRL H} {CTRL J}{CTRL H}{CTRL H
    } {CTRL J}{CTRL H}{CTRL H} {CTRL J}{
    CTRL H}{CTRL H} {CTRL J}{CTRL H}{CTR
    L H} {CTRL K}{CTRL K}{CTRL K}{CTRL K
    }{CTRL K}{CTRL K}{CTRL K}{CTRL K}{CT
    RL K}{CTRL K}{CTRL K}{CTRL K}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I}{CTRL I}{CTRL I}{CTRL I}{CT
    RL I}{CTRL I}{CTRL I}{CTRL I}{CTRL I
    }{CTRL I} {CTRL J} {CTRL J} {CTRL J}
     {CTRL J} {CTRL J} {CTRL J} {CTRL J}
     {CTRL J} {CTRL J} {CTRL J} {CTRL J}
     {CTRL J} {CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL K}{CTRL K}{CTRL K}{CTRL
    K}{CTRL K}{CTRL K}{CTRL K}{CTRL K}{C
    TRL K}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}{C
    TRL H}{CTRL H}{CTRL H}{CTRL H}{CTRL
    H}{CTRL H}{CTRL H}{CTRL H}{CTRL H}"  [4A1E]
500 z$(1)=STRING$(40,CHR$(143))          [AFB0]
510 z$(2)=CHR$(143)+CHR$(129)+SPACE$(17)
    +CHR$(139)+CHR$(135)+SPACE$(17)+CHR$
    (130)+CHR$(143)                      [74AA]
520 z$(3)=CHR$(143)+SPACE$(18)+CHR$(138)
    +CHR$(133)+SPACE$(18)+CHR$(143)      [0C58]
530 z$(4)=z$(3):z$(5)=z$(3):z$(6)=z$(3):
    z$(7)=z$(3):z$(8)=z$(3):z$(9)=z$(3):
    z$(10)=z$(3):z$(11)=z$(3):z$(12)=z$(
    3):z$(13)=z$(3)                      [BF3E]
540 z$(14)=CHR$(143)+CHR$(132)+SPACE$(17
    )+CHR$(142)+CHR$(141)+SPACE$(17)+CHR
    $(136)+CHR$(143)                     [0904]
550 z$(15)=STRING$(40,CHR$(143))         [0824]
560 z$(16)=SPACE$(40)                    [0326]
570 z$(17)=CHR$(32)+CHR$(150)+CHR$(154)+
    "HGS"+CHR$(154)+CHR$(154)+CHR$(156)+
    CHR$(150)+CHR$(154)+CHR$(154)+"KOM"+
    CHR$(154)+CHR$(156)+SPACE$(7)++CHR$(
    150)+CHR$(154)+"LDB"+CHR$(154)+CHR$(
    156)+CHR$(32)+CHR$(134)+CHR$(132)+CH
    R$(32)+CHR$(208)                     [4538]
580 z$(17)=z$(17)+CHR$(213)+CHR$(143)+CH
    R$(143)+CHR$(32)                     [258E]
590 z$(18)=CHR$(32)+CHR$(149)+SPACE$(6)+
    "k"+CHR$(149)+SPACE$(6)+CHR$(255)+CH
    R$(150)+"HRZ"+CHR$(156)+SPACE$(2)+"V
    "+SPACE$(5)+"V"+SPACE$(4)+"0"+CHR$(1
    50)+"K"+CHR$(156)+CHR$(32)           [2BEC]
600 z$(19)=CHR$(32)+CHR$(147)+STRING$(6,
    CHR$(154))+CHR$(153)+CHR$(147)+STRIN
    G$(6,CHR$(154))+CHR$(153)+CHR$(149)+
    CHR$(32)+CHR$(159)+CHR$(32)+CHR$(149
    )+CHR$(150)+CHR$(154)+CHR$(155)+CHR$
    (154)+CHR$(154)+CHR$(158)+CHR$(154)+
    CHR$(154)+CHR$(155)+CHR$(154)        [2ED2]
610 z$(19)=z$(19)+CHR$(156)+SPACE$(3)+CH
    R$(149)+CHR$(32)+CHR$(149)+CHR$(32)  [1FF0]
620 z$(20)=CHR$(32)+CHR$(150)+CHR$(154)+
    "VGS"+CHR$(154)+CHR$(154)+CHR$(156)+
    CHR$(150)+CHR$(154)+CHR$(154)+SPACE$
    (3)+CHR$(154)+CHR$(156)+CHR$(149)+CH
    R$(32)+CHR$(159)+CHR$(32)+CHR$(149)+
    "L"+SPACE$(4)+CHR$(149)+SPACE$(4)+"R
    "+CHR$(32)+CHR$(150)+"MOT"           [B3FC]
630 z$(20)=z$(20)+CHR$(157)+CHR$(32)     [EE28]
640 z$(21)=CHR$(32)+CHR$(149)+SPACE$(6)+
    "f"+CHR$(149)+SPACE$(6)+CHR$(255)+CH
    R$(151)+CHR$(32)+CHR$(159)+CHR$(32)+
    CHR$(157)+CHR$(147)+CHR$(154)+CHR$(1
    58)+CHR$(154)+CHR$(154)+CHR$(155)+CH
    R$(154)+CHR$(154)+CHR$(158)+CHR$(154
    )+CHR$(153)+CHR$(32)+CHR$(149)       [4574]
650 z$(21)=z$(21)+STRING$(3,CHR$(230))+C
    HR$(149)+CHR$(32)                    [21AC]
660 z$(22)=CHR$(32)+CHR$(151)+CHR$(154)+
    "H*H"+CHR$(154)+CHR$(154)+CHR$(157)+
    CHR$(151)+CHR$(154)+CHR$(154)+"ENT"+
    CHR$(154)+CHR$(157)+CHR$(149)+CHR$(3
    2)+CHR$(159)+CHR$(32)+CHR$(149)+SPAC
    E$(2)+"H"+SPACE$(5)+"H"+CHR$(150)+CH
    R$(154)+CHR$(154)+"TR"+CHR$(154)     [5BB0]
670 z$(22)=z$(22)+CHR$(154)+CHR$(157)+CH
    R$(32)                               [1A66]
680 z$(23)=CHR$(32)+CHR$(149)+SPACE$(6)+
    "f"+CHR$(149)+SPACE$(6)+"m"+CHR$(149
    )+CHR$(32)+CHR$(159)+CHR$(32)+CHR$(1
    49)+SPACE$(2)+CHR$(147)+STRING$(5,CH
    R$(154))+CHR$(153)+CHR$(149)+SPACE$(
    6)+"1"+CHR$(32)                      [DBA4]
690 z$(24)=CHR$(32)+CHR$(147)+STRING$(6,
    CHR$(154))+CHR$(153)+CHR$(147)+STRIN
    G$(6,CHR$(154))+CHR$(153)+CHR$(147)+
    STRING$(3,CHR$(154))+CHR$(153)++SPAC
    E$(9)+CHR$(147)+STRING$(6,CHR$(154))
    +CHR$(153)+CHR$(32)                  [3B4E]
700 z$(25)=SPACE$(40)                    [E01E]
800 CLS:PRINT"{12 SPACE}JETLINER"        [7604]
810 PRINT"{2 SPACE}For VC-20 by Roman Gr
    andis (CP 2/85)"                     [AC56]
820 PRINT"{4 SPACE}Adapted for CPC by Cl
    aus Herwig"                          [C268]
830 PRINT                                [7690]
840 PRINT"Hamburg...........1"           [D5A0]
850 PRINT"Bremen............2"           [AF26]
860 PRINT"Hannover..........3"           [0242]
870 PRINT"M;nchen...........4"           [F948]
880 PRINT"K:ln/Bonn.........5"           [DAA2]
890 PRINT"Frankfurt.........6"           [E1D6]
900 PRINT"N;rnberg..........7"           [19D0]
910 PRINT"Stuttgart.........8"           [3D0A]
920 PRINT"D;sseldorf........9"           [A8D0]
930 PRINT"Berlin/Tegel......0"           [243E]
940 PRINT"Husum.............U"           [C03A]
950 PRINT"Leck..............L"           [F960]
960 PRINT"Eggebek...........E"           [4396]
970 PRINT"Schleswig.........S"           [0FFA]
980 PRINT"Bremgarten........B"           [AF3A]
990 PRINT"F;rstenfeldbruck.F"            [BFCC]
1000 PRINT"Hopsten..........O"           [685C]
1010 PRINT"Memmingen........M"           [341A]
1020 PRINT"Remstein.........R"           [F1D6]
1030 PRINT:PRINT"{3 SPACE}Bitte w@hlen"  [AAA4]
1040 a$=INKEY$                           [D696]
1050 a$=LOWER$(a$)                       [6356]
1060 IF a$="1" THEN a1=0.25:a2=0:a3=310:
     bn1$="ALF":GOTO 5060                [1574]
1070 IF a$="2" THEN a1=0.16:a2=0:a3=230:
     bn1$="BMN":GOTO 5060                [8C8E]
1080 IF a$="3" THEN a1=0.1:a2=0:a3=255:b
     n1$="HAD":GOTO 5060                 [5014]
1090 IF a$="4" THEN a1=-0.34:a2=0:a3=160
     :bn1$="DM ":GOTO 5060               [119C]
1100 IF a$="5" THEN a1=-0.62:a2=0:a3=140
     :bn1$="KBO":GOTO 5060               [98E2]
1110 IF a$="6" THEN a1=0.05:a2=0:a3=160:
     bn1$="DF ":GOTO 5060                [9426]
1120 IF a$="7" THEN a1=0.6:a2=0:a3=350:b
     n1$="NUB":GOTO 5060                 [A044]
1130 IF a$="8" THEN a1=-0.4:a2=0:a3=160:
     bn1$="DS ":GOTO 5060                [4F40]
1140 IF a$="9" THEN a1=0.3:a2=0:a3=290:b
     n1$="DUS":GOTO 5060                 [CF5A]
1150 IF a$="0" THEN a1=0.2:a2=0:a3=350:b
     n1$="TGL":GOTO 5060                 [9838]
1160 IF a$="u" THEN a1=-0.2:a2=0:a3=125:
     bn1$="HUU":GOTO 5060                [B234]
1170 IF a$="l" THEN a1=-0.36:a2=0:a3=30:
     bn1$="LCK":GOTO 5060                [3BF8]
1180 IF a$="e" THEN a1=0.2:a2=0:a3=280:b
     n1$="EGB":GOTO 5060                 [067A]
1190 IF a$="s" THEN a1=0.1:a2=0:a3=160:b
     n1$="SWG":GOTO 5060                 [53D6]
1200 IF a$="b" THEN a1=-0.7:a2=0:a3=320:
     bn1$="BGT":GOTO 5060                [D1DE]
1210 IF a$="f" THEN a1=0.4:a2=0:a3=360:b
     n1$="FFB":GOTO 5060                 [1572]
```




```
1220 IF a$="o" THEN a1=-0.18:a2=0:a3=100
     :bn1$="HOP":GOTO 5060                [786C]
1230 IF a$="m" THEN a1=-0.45:a2=0:a3=330
     :bn1$="MEM":GOTO 5060                [9364]
1240 IF a$="r" THEN a1=0.7:a2=0:a3=180:b
     n1$="RMS":GOTO 5060                  [4BDE]
1250 GOTO 1040                            [550C]
1300 MODE 1:a4=1                          [B22E]
1310 PEN 2                                [E934]
1320 FOR i=1 TO 25                        [CA1C]
1330 PRINT z$(i);                         [284E]
1340 NEXT i                               [CC60]
1350 LOCATE wmot,smot:PEN 3:PRINT STRING
     $(3,CHR$(230))                       [35C0]
1370 LOCATE wff,sff:PEN 1:PRINT bn$       [8C74]
1380 LOCATE wkr,skr:PEN 1:PRINT"0"        [DC7A]
1390 FOR i=1 TO 12:PLOT 289+i,31:DRAWR 0
     ,40:NEXT                             [7008]
1400 FOR i=1 TO 12:PLOT 321+i,31:DRAWR 0
     ,40:NEXT                             [31DE]
1410 b3=20:b4=1:k8=1:b5=0:b6=0:b7=0:b8=1
     6:d1=0:d2=1:n7=30000:sq1=0:d4$="UNT
     EN"                                  [ADB2]
1420 l7=0:d6=1:t1=TIME/300                [688C]
1440 dl=0:k2=0:zf=1                       [E3E4]
1450 EVERY 15,1 GOSUB 6600                [E3C8]
2000 DI                                   [50DE]
2010 b=JOY(0)                             [91EA]
2020 b$=INKEY$                            [DF96]
2030 b$=LOWER$(b$)                        [5658]
2040 IF ww=1 THEN 2120                    [0A4A]
2050 IF b=1 OR b$="i" THEN b3=b3+d8:GOTO
      2260                                [8262]
2060 IF b=2 OR b$="m" THEN b3=b3-d8::ru=
     1:GOTO 2260                          [2804]
2070 IF b=4 OR b$="j" THEN b5=b5-1:GOTO
     2260                                 [2EA4]
2080 IF b=8 OR b$="k" THEN b5=b5+1:GOTO
     2260                                 [D5AC]
2090 EI                                   [5FF2]
2100 IF b=16 OR b$="z" THEN 5000          [C45C]
2110 DI                                   [5DE2]
2120 IF b$="q" THEN ti1=TIME/300:ww=1     [639A]
2130 IF b$="w" THEN ti2=TIME/300:ww=0:t1
     =t1-(ti2-ti1)                        [0FBA]
2140 IF ww=1 THEN 2020                    [6C4A]
2150 IF b$="]" THEN e1=e1-1:IF e1<1 THEN
      e1=0                                [791E]
2160 IF b$="[" THEN e1=e1+1:IF e1>3 THEN
      e1=4                                [C528]
2170 IF b$="a" THEN b4=0                  [2DE6]
2180 IF b$="s" THEN b4=1                  [CF0E]
2190 IF b$="\" THEN h2=h2+180:GOTO 2700   [A1FC]
2200 IF b6=1  AND d6=1 THEN t1=TIME/300:
     d6=0                                 [39AE]
2210 IF b$="" THEN 2260                   [80A8]
2220 IF ASC(b$)<48 OR ASC(b$)>57 THEN 22
     60                                   [43BA]
2230 d1=ASC(b$)-48:d1$=b$:IF d1=0 THEN d
     1$="B"                               [6518]
2240 IF eo=1 THEN d1=d1-3:IF d1<0 THEN d
     1=0                                  [83D4]
2250 IF f3=0 AND d1<>1 THEN d1=0:d1$="0"
                                          [010C]
2255 IF n7=0 THEN d1=0                    [0F40]
2260 LOCATE wkr,skr                       [9E24]
2270 PEN 1:PRINT d1$                      [7B7E]
2280 EI                                   [F5F4]
2290 IF d1$="B" THEN n7=n7-0.1            [7194]
2300 IF b6=0 THEN n7=n7-5*d1              [8E60]
2320 n7=n7-(d1-(d1*b8/60000))*e6          [7A84]
2330 IF n7<=0 THEN n7=0:LOCATE wmot,smot
     :PEN 3:INK 3,6:PRINT STRING$(3,CHR$
     (230)):d1=0:d1$="0":GOTO 2340        [A548]
2340 DI                                   [69EC]
2350 LOCATE wtr,str:IF n7>3000 OR n7=0 T
     HEN PRINT"{CTRL H} ";                [9E92]
2360 PEN 1:PRINT USING"#####";n7          [3DBE]
2370 GOSUB 6400                           [BFAC]
2380 EI                                   [52F6]
2390 e6=(TIME/300-e7):IF e6<zf THEN 2390
                                          [165C]
2392 IF e6>zf THEN e6=zf                  [6AFE]
2394 e7=TIME/300                          [3878]
2400 IF f1<128 THEN f1=128                [F0C2]
2410 IF f3=0 AND d1=1 AND d1=1 THEN GOSU
     B 6000:f3=1                          [2432]
2420 ef=INT(5000*RND(1))+1                [D7BA]
2430 IF ef<>34 OR eo=1 OR b6=0 THEN 2480
                                          [9814]
2440 DI                                   [4CEE]
2450 LOCATE wmot+2,smot:PEN 2:PRINT CHR$
     (231)                                [4928]
2460 EI                                   [7FF4]
2470 eo=1                                 [855E]
2480 f4=(d1*0.9-b7/70-(20-b3)/5-e1/2-b4)
     *e6:b7=b7+f4:IF b7<0 THEN b7=0       [D996]
2490 f5=b7:IF b7>5 THEN a3=a3+b5*e6       [1860]
2500 IF a3>360 THEN a3=a3-360             [6C36]
2510 IF a3<0 THEN a3=a3+360               [375E]
2520 DI                                   [33EC]
2530 LOCATE wkom,skom                     [59CC]
2540 PEN 1:PRINT USING"####.#";a3         [EEF8]
2550 EI                                   [80F4]
2560 f8=b7*e6/3600                        [B93C]
2570 h1=a3*PI/180:a1=a1+f8*SIN(h1)        [7C8A]
2580 a2=a2+f8*COS(h1)                     [8628]
2590 IF a2=0 THEN a2=0.001                [189A]
2600 h3=SQR(a1^2+a2^2)                    [E8EE]
2610 h4=ABS((ATN(a1/a2))/PI*180):h5=h6+a
     1:h7=h8+a2                           [B260]
2620 IF h5>h6 AND h7=h8 THEN h2=270       [D600]
2630 IF h5<h6 AND h7=h8 THEN h2=90        [279E]
2640 IF h5=h6 AND h7>h8 THEN h2=180       [CF04]
2650 IF h5=h6 AND h7<h8 THEN h2=360       [D302]
2660 IF h5>h6 AND h7>h8 THEN h2=180+h4    [F798]
2670 IF h5>h6 AND h7<h8 THEN h2=360-h4    [E09A]
2680 IF h5<h6 AND h7>h8 THEN h2=180-h4    [CB9C]
2690 IF h5<h6 AND h7<h8 THEN h2=360+h4    [F496]
2700 IF h2>360 THEN h2=h2-360             [955E]
2710 IF h2<0 THEN h2=h2+360               [8286]
2720 DI                                   [59F0]
2730 LOCATE wkkom,skkom                   [747C]
2740 PEN 1:PRINT USING"####.#";h2         [F408]
2750 EI:DI                                [C086]
2760 DI:LOCATE wkent,skent                [A610]
2770 PEN 1:PRINT USING"####.#";h3         [8810]
2780 EI                                   [5EFE]
2790 k2=(b3-20)*-b7+(e1-0.5*b4)*150:k3=b
     7*(1+0.2*e1)                         [21B0]
2800 IF b3=20 THEN k2=0                   [5E8C]
2810 IF (b8-16)=0 AND b3>20 THEN b3=20    [5858]
2820 IF k3<180 THEN k2=k2-(180-k3)^2      [4CCC]
2830 IF b6=0 AND k2<0 THEN k2=0           [816C]
2840 IF b8>32000 AND k2>0 THEN k2=k2*((4
     2000-b8)/10000)                      [922E]
2850 IF b8<(16+300) AND k2<0 AND b4=0 TH
     EN INK 3,1,24:sd=1                   [2CD2]
2860 IF b7>500 THEN INK 3,1,24:sd=1:DI:L
     OCATE wwl2,swl2:PEN 3:PRINT CHR$(23
     1):EI                                [757A]
2862 IF k2<-8000 THEN INK 3,1,24:sd=1:DI
     :LOCATE wwl1,swl1:PEN 3:PRINT CHR$(
     231):EI                              [6F3E]
2865 IF n7<3000 AND n7>0 THEN INK 3,1,24
     :DI:LOCATE wwl3,swl3:PEN 3:PRINT CH
     R$(231):EI                           [4946]
2866 IF sd=1 THEN sd=0:SOUND 2,284,50,1   [CF4A]
2910 DI                                   [15F2]
2915 IF k2<-9999 THEN k2=-9999            [7DC8]
2920 LOCATE wvgs,svgs:IF k2>-8000 THEN P
     RINT"{CTRL H} ";                     [81AE]
2930 PEN 1:PRINT USING"#####";k2          [01B4]
2940 EI                                   [5CFA]
2950 d8=3:IF b3>10 AND b3<30 THEN d8=2    [CBB4]
2960 IF b3>14 AND b3<26 THEN d8=1         [523A]
2970 b8=b8+e6*k2/60:IF(b8-16)<0 THEN b8=
     16:b6=0                              [F936]
2980 IF (b8-16)>2 THEN b6=1               [F682]
2990 IF b6=0 AND d1<1 THEN b7=b7-4        [D4F8]
3010 DI:LOCATE whoe,shoe                  [8622]
3020 IF (b8-16)>300 THEN PEN 1:PRINT USI
     NG"######";b8:GOTO 3030              [72E6]
3025 IF (b8-16)<300 AND b4=0 AND k2<0 TH
     EN PEN 3 ELSE PEN 2                  [762E]
3026 PRINT"H";:PEN 1:PRINT USING"#####";
     (b8-16)                              [AA04]
3030 EI                                   [81E8]
3040 IF e1=k6 THEN 3120                   [3C7E]
3050 SOUND 2,568,100,1                    [6ACE]
3060 k6=e1:DI:LOCATE wkl,skl              [EAF4]
3070 IF k6=0 THEN PEN 2:PRINT CHR$(208)
     ELSE PEN 1:PRINT CHR$(204)           [5F14]
3080 EI:DI                                [E080]
3090 LOCATE wkln,skln                     [F0C8]
3100 IF k6=0 THEN PEN 2:PRINT USING"#";k
     6 ELSE PEN 1:PRINT USING"#";k6       [D484]
3110 EI                                   [40E6]
3120 IF k8=b4 THEN 3200                   [537E]
3130 SOUND 2,1136,100,1                   [BD1C]
3140 k8=b4                                [EBBC]
3150 DI                                   [57EC]
3160 LOCATE wfw,sfw                       [A724]
3170 IF k8=0 THEN PEN 2:PRINT CHR$(137)C
     HR$(129):d4$=" OBEN"                 [A796]
3180 IF k8=1 THEN k9=RND(1):IF k9>0.8 TH
     EN PEN 2:PRINT CHR$(134)CHR$(32):d4
     $="{2 SPACE}?{2 SPACE}" ELSE PEN 2:
     PRINT CHR$(134)CHR$(132):d4$="UNTEN
     "                                    [B5FA]
3190 EI                                   [50F6]
3200 IF b7<0 THEN b7=0                    [401C]
3210 DI:LOCATE wgs,sgs:IF b7<500 THEN PR
     INT"{CTRL H} ";                      [268A]
3220 PEN 1:PRINT USING"#####";b7:EI       [5B2E]
3230 IF (b8-16)>300 AND ktrl=1 THEN 3440
                                          [5234]
3240 PEN 1:DI                             [B7C8]
3250 LOCATE wlav,slav                     [EEBC]
3260 IF (b8-16)>300 THEN PRINT"{5 SPACE}
     ":GOTO 3290                          [F3CC]
```

Listing. Fliegen ohne Pilotenschein (Fortsetzung)

```
3270 IF ABS(er-h5)<10 AND a3<=180 THEN P
     RINT USING"#####";((er-h5)*1000):GO
     TO 3290                              [9530]
3280 IF ABS(h5-fr)<10 AND a3>180 THEN PR
     INT USING"#####";((h5-fr)*1000):GOT
     O 3290                               [07C0]
3285 PRINT"{5 SPACE}"                     [EDC6]
3290 LOCATE wlah,slah                     [048C]
3300 IF (b8-16)>300 THEN PRINT"{5 SPACE}
     ":GOTO 3350                          [88BC]
3310 IF ABS(h5-fr)<10 AND a3<=180 THEN P
     RINT USING"#####";((h5-fr)*1000):GO
     TO 3350                              [9324]
3320 IF ABS(er-h5)<10 AND a3>180 THEN PR
     INT USING"#####";((er-h5)*1000):GOT
     O 3350                               [DBAC]
3325 PRINT"{5 SPACE}"                     [4CBC]
3330 EI                                   [3EEE]
3340 DI                                   [4BEE]
3350 LOCATE wlal,slal                     [8D96]
3360 IF (b8-16)>300 THEN PRINT"{4 SPACE}
     ":GOTO 3390                          [8490]
3370 IF ABS(h7-hr)<1 AND a3>=180 THEN PR
     INT USING"####";((h7-hr)*1000):GOTO
      3390                                [54A6]
3380 IF ABS(gr-h7)<1 AND a3<180 THEN PRI
     NT USING"####";((gr-h7)*1000):GOTO
     3390                                 [3426]
3385 PRINT"{4 SPACE}"                     [1E88]
3390 LOCATE wlar,slar                     [D7B6]
3400 IF (b8-16)>300 THEN PRINT"{4 SPACE}
     ":ktrl=1:GOTO 3440                   [2448]
3410 ktrl=0                               [1E24]
3420 IF ABS(gr-h7)<1 AND a3>=180 THEN PR
     INT USING"####";((gr-h7)*1000):GOTO
      3440                                [9A92]
3430 IF ABS(h7-hr)<1 AND a3<180 THEN PRI
     NT USING"####";((h7-hr)*1000):GOTO
     3440                                 [B11A]
3435 PRINT"{4 SPACE}"                     [BF80]
3440 EI:IF e1=1 AND b7>230 THEN 17=1:GOT
     O 4000                               [9D78]
3450 IF e1=2 AND b7>210 THEN 17=2:GOTO 4
     000                                  [B0EA]
3460 IF e1=3 AND b7>190 THEN 17=3:GOTO 4
     000                                  [D5FE]
3470 IF e1=4 AND b7>170 THEN 17=4:GOTO 4
     000                                  [2900]
3480 IF b4=1 AND b7>270 THEN 17=5:GOTO 4
     000                                  [3D00]
3490 IF (b8-16)=0 AND k2<-500 THEN 17=6:
     GOTO 4000                            [60A0]
3500 IF b7>520 THEN 17=7:GOTO 4000        [A3C4]
3510 IF b6=0 AND(h5>(er+0.1) OR h5<(fr-0
     .1) OR h7>(gr+0.1) OR h7<(hr-0.1))
     THEN 17=8:GOTO 4000                  [29C0]
3520 IF b6=0 AND(h5>er OR h5<fr OR h7>gr
      OR h7<hr) THEN 17=9:GOTO 4000       [0264]
3525 IF (b8-16)=0 AND k2<-500 THEN 17=6:
     GOTO 4000                            [589E]
3530 IF b6=0 AND d4$=" OBEN" THEN 17=10:
     GOTO 4000                            [FF74]
3540 IF b6=0 AND d4$="{2 SPACE}?{2 SPACE
     }" THEN 17=11:GOTO 4000              [206E]
3550 IF b6=0 AND b7>190 THEN 17=12:GOTO
     4000                                 [315C]
3560 IF b6=0 AND b7>60 AND b5<>0 THEN 17
     =13:GOTO 4000                        [42A0]
3565 IF k2<-9900 THEN 17=14:GOTO 4000     [A40A]
3570 IF b6=0 THEN b3=20                   [628C]
3580 IF b7<1 AND b6=0 AND d6=0 THEN t2=T
     IME/300:GOTO 4300                    [22E8]
3590 dl=1                                 [BB5E]
3600 GOTO 2000                            [9F08]
4000 t2=TIME/300                          [4870]
4010 ENV 1,5,0,100,5,-3,100               [D086]
4020 PEN 0:INK 0,1,24:LOCATE 10,3:PRINT
     REMAIN(1)                            [3DD6]
4030 SOUND 2,119,900,15,1,,10             [73EC]
4040 FOR i=1 TO 10000:NEXT                [3626]
4050 INK 0,1                              [92F0]
4060 PEN 1:LOCATE 10,7:PRINT"ENTER"       [6FFA]
4070 IF INKEY(18)=-1 THEN 4070            [0752]
4080 CLS                                  [C69C]
4090 MODE 2:PRINT"{3 SPACE}Unfallursache
     ":PRINT                              [ECC6]
4110 IF 17=1 THEN PRINT"Sie haben mit La
     ndeklappen'1' die H:chstgeschwindig
     keit von 230 k ;berschritten"        [C9EE]
4120 IF 17=2 THEN PRINT"Sie haben mit La
     ndeklappen'2' die H:chstgeschwindig
     keit von 210 k ;berschritten"        [02F0]
4130 IF 17=3 THEN PRINT"Sie haben mit La
     ndeklappen'3' die H:chstgeschwindig
     keit von 190 k ;berschritten"        [FD04]
4140 IF 17=4 THEN PRINT"Sie haben mit La
     ndeklappen'4' die H:chstgeschwindig
     keit von 170 k ;berschritten"        [4406]
4150 IF 17=5 THEN PRINT"Sie haben mit au
     sgefahrenem Fahrwerk die H:chstgesc
     hwindigkeit von 270 k ;ber-{2 SPACE
     }schritten"                          [D2AE]
4160 IF 17=12 THEN PRINT"Sie haben die H
     :chstgeschwindigkeit am Boden von 1
     90 k ;berschritten"                  [2FA2]
4170 IF 17=7 THEN PRINT"Sie haben die H:
     chstgeschwindigkeit von 520 k ;bers
     chritten"                            [045A]
4180 IF 17=11 THEN PRINT"Sie haben das r
     echte Fahrwerk nicht ausgefahren"    [F312]
4190 IF 17=6 THEN PRINT"Sie haben zu har
     t aufgesetzt"                        [80E4]
4200 IF 17=10 THEN PRINT"Sie haben das H
     auptfahrwerk nicht ausgefahren"      [2810]
4210 IF 17=13 THEN PRINT"Sie haben bei z
     u hoher Rollgeschwindigkeit gekurvt
     "                                    [8310]
4220 IF 17=8 THEN PRINT"Sie haben au^erh
     alb der Landebahn aufgesetzt"        [7A94]
4230 IF 17=9 THEN PRINT"Sie sind von der
      Runway abgekommen"                  [9F14]
4240 IF 17=14 THEN PRINT"Sie haben die m
     aximale Vertikalgeschwindigkeit von
      -9900 f ;berschritten"              [A570]
4250 t3=t2-t1:st=INT(t3/3600):mi=INT(t3/
     60-st*60):sk=INT(t3-mi*60-st*3600)   [9476]
4260 PRINT:PRINT"Flugzeit "st" St "mi" M
     in "sk" Sek"                         [4206]
4270 PRINT"Treibstoffverbrauch ";:trv=tr
     v+(30000-n7):PRINT INT(trv)"l"       [1824]
4280 PRINT:PRINT"N@chster Flug f, sonst
     n"                                   [31C4]
4290 a$=INKEY$:IF a$="" THEN 4290         [3D04]
4293 IF a$="n" THEN END                   [03B6]
4295 IF a$<>"f" THEN 4290 ELSE RUN        [41D0]
4300 IF ru=0 THEN 2000                    [1136]
4310 ti1=TIME/300:IF ru=1 THEN ru=0       [7C4A]
4320 d$=INKEY$                            [E1A4]
4330 IF d$="t" THEN eo=0:DI:LOCATE wmot,
     smot:sou=1:PEN 1:PRINT STRING$(3,CH
     R$(231)):EI:trv=trv+(30000-n7):n7=3
     0000:ti2=TIME/300:t1=t1-(ti2-ti1):G
     OTO 2000                             [C8B0]
4340 IF d$="e" THEN t2=TIME/300:GOTO 425
     0                                    [F452]
4350 GOTO 4310                            [621A]
5000 IF b6=0 THEN 2000                    [4B94]
5001 DI:LOCATE wff,sff                    [4966]
5010 PRINT"{3 SPACE}":EI                  [2CBE]
5020 PEN 2                                [D238]
5030 a$=INKEY$:IF a$="" THEN 5030 ELSE a
     $=UPPER$(a$):bn1$=a$:DI:PEN 2:LOCAT
     E wff,sff:PRINT bn1$:EI              [5920]
5040 b$=INKEY$:IF b$="" THEN 5040 ELSE b
     $=UPPER$(b$):bn1$=bn1$+b$:DI:PEN 2:
     LOCATE wff,sff:PRINT bn1$:EI         [95CE]
5050 c$=INKEY$:IF c$="" THEN 5050 ELSE c
     $=UPPER$(c$):bn1$=bn1$+c$:DI:PEN 2:
     LOCATE wff,sff:PRINT bn1$:EI         [EEDC]
5060 IF bn1$="ALF" THEN zz=53:GOSUB 5760
     :h8=1218.2:h6=355.9:GOSUB 5700:er=3
     56.2:fr=355:GOTO 5720                [F6C4]
5070 IF bn1$="ALS" THEN h8=1295:h6=345:G
     OSUB 5700:GOTO 5720                  [8DEC]
5080 IF bn1$="ALB" THEN h8=953:h6=440:GO
     SUB 5700:GOTO 5720                   [BB64]
5090 IF bn1$="BMN" THEN zz=11:GOSUB 5760
     :h8=1182.7:h6=317.1:GOSUB 5700:er=3
     18.1:fr=316.6:GOTO 5720              [0884]
5100 IF bn1$="BAM" THEN h8=1080:h6=269:G
     OSUB 5700:GOTO 5720                  [77BA]
5110 IF bn1$="BGT" THEN zz=700:GOSUB 576
     0:h8=874.4:h6=307.2:GOSUB 5700:er=3
     07.4:fr=306.2:GOTO 5720              [9A80]
5120 IF bn1$="COL" THEN h8=1047:h6=288:G
     OSUB 5700:GOTO 5720                  [FEE2]
5130 IF bn1$="DKB" THEN h8=949:h6=402:GO
     SUB 5700:GOTO 5720                   [4C66]
5140 IF bn1$="DOM" THEN h8=1103:h6=282:G
     OSUB 5700:GOTO 5720                  [40D0]
5150 IF bn1$="DUS" THEN zz=147:GOSUB 576
     0:h8=1077:h6=253.8:GOSUB 5700:er=25
     4.8:fr=253.4:GOTO 5720               [B060]
5160 IF bn1$="DLE" THEN h8=1135:h6=363:G
     OSUB 5700:GOTO 5720                  [AEC8]
5170 IF bn1$="DHE" THEN h8=1251:h6=278:G
     OSUB 5700:GOTO 5720                  [08CA]
5180 IF bn1$="DM " THEN zz=1737:GOSUB 57
     60:h8=887.9:h6=469:GOSUB 5700:er=47
     0:fr=468.3:GOTO 5720                 [A262]
5190 IF bn1$="DF " THEN zz=368:GOSUB 576
     0:h8=1002.3:h6=330.7:GOSUB 5700:er=
     331.5:fr=329.6:GOTO 5720             [998C]
5200 IF bn1$="DS " THEN zz=1300:GOSUB 57
     60:h8=921.3:h6=365.2:GOSUB 5700:er=
     366:fr=364.4:GOTO 5720               [43D8]
5210 IF bn1$="EGB" THEN zz=65:GOSUB 5760
     :h8=1277.4:h6=324.9:GOSUB 5700:er=3
     25.4:fr=324.5:GOTO 5720              [667A]
5220 IF bn1$="ERD" THEN h8=900:h6=477:GO
     SUB 5700:GOTO 5720                   [EF78]
5230 IF bn1$="ERL" THEN h8=979:h6=434:GO
     SUB 5700:GOTO 5720                   [E49C]
5240 IF bn1$="EEL" THEN h8=1189:h6=240:G
     OSUB 5700:GOTO 5720                  [81CE]
```

```
5250 IF bn1$="FFM" THEN h8=1003:h6=333:G
     OSUB 5700:GOTO 5720                      [54BE]
5260 IF bn1$="FFB" THEN zz=1703:GOSUB 57
     60:h8=892.6:h6=451.2:GOSUB 5700:er=
     451.6:fr=450.7:GOTO 5720                 [C9E8]
5270 IF bn1$="FUL" THEN h8=1036:h6=365:G
     OSUB 5700:GOTO 5720                      [0EF4]
5280 IF bn1$="GED" THEN h8=1025:h6=354:G
     OSUB 5700:GOTO 5720                      [B2C0]
5290 IF bn1$="GMH" THEN h8=1070:h6=297:G
     OSUB 5700:GOTO 5720                      [F5E6]
5300 IF bn1$="GSO" THEN h8=1116:h6=307:G
     OSUB 5700:GOTO 5720                      [B9E2]
5310 IF bn1$="HAM" THEN h8=1221:h6=363:G
     OSUB 5700:GOTO 5720                      [92BC]
5320 IF bn1$="HAD" THEN zz=183:GOSUB 576
     0:h8=1147.7:h6=354.6:GOSUB 5700:er=
     355.3:fr=353.7:GOTO 5720                 [A1E8]
5330 IF bn1$="HOP" THEN zz=129:GOSUB 576
     0:h8=1140.8:h6=276.7:GOSUB 5700:er=
     277:fr=276.4:GOTO 5720                   [8460]
5340 IF bn1$="HUU" THEN zz=93:GOSUB 5760
     :h8=1270.9:h6=318.6:GOSUB 5700:er=3
     19.4:fr=318.4:GOTO 5720                  [0FD2]
5350 IF bn1$="INN" THEN h8=836:h6=465:GO
     SUB 5700:GOTO 5720                       [DD9E]
5360 IF bn1$="KBO" THEN zz=300:GOSUB 576
     0:h8=1051.8:h6=270.9:GOSUB 5700:er=
     271.6:fr=270.2:GOTO 5720                 [D8E4]
5370 IF bn1$="LBU" THEN h8=935:h6=372:GO
     SUB 5700:GOTO 5720                       [DD98]
5380 IF bn1$="LBE" THEN h8=1219:h6=342:G
     OSUB 5700:GOTO 5720                      [29CC]
5390 IF bn1$="LCK" THEN zz=24:GOSUB 5760
     :h8=1287.6:h6=309.7:GOSUB 5700:er=3
     10.9:fr=309.1:GOTO 5720                  [A39E]
5400 IF bn1$="LUX" THEN h8=977:h6=241:GO
     SUB 5700:GOTO 5720                       [31BA]
5410 IF bn1$="LNZ" THEN h8=893:h6=564:GO
     SUB 5700:GOTO 5720                       [FEBC]
5420 IF bn1$="LNO" THEN h8=1036:h6=218:G
     OSUB 5700:GOTO 5720                      [5BEC]
5430 IF bn1$="MEM" THEN zz=2079:GOSUB 57
     60:h8=879.5:h6=412:GOSUB 5700:er=41
     2.3:fr=410.9:GOTO 5720                   [5448]
5440 IF bn1$="MAH" THEN h8=896:h6=452:GO
     SUB 5700:GOTO 5720                       [7F84]
5450 IF bn1$="MTR" THEN h8=1017:h6=340:G
     OSUB 5700:GOTO 5720                      [62FC]
5460 IF bn1$="MUN" THEN h8=891:h6=473:GO
     SUB 5700:GOTO 5720                       [3EB8]
5470 IF bn1$="NTM" THEN h8=1001:h6=252:G
     OSUB 5700:GOTO 5720                      [8CEE]
5480 IF bn1$="NIE" THEN h8=1158:h6=342:G
     OSUB 5700:GOTO 5720                      [FBE4]
5490 IF bn1$="NOR" THEN h8=1051:h6=254:G
     OSUB 5700:GOTO 5720                      [9D00]
5500 IF bn1$="NDO" THEN h8=1226:h6=307:G
     OSUB 5700:GOTO 5720                      [45DA]
5510 IF bn1$="NUB" THEN zz=1045:GOSUB 57
     60:h8=971.5:h6=431.5:GOSUB 5700:er=
     433:fr=431.3:GOTO 5720                   [F73A]
5520 IF bn1$="OBG" THEN h8=1191:h6=293:G
     OSUB 5700:GOTO 5720                      [ADD6]
5530 IF bn1$="OSN" THEN h8=1132:h6=305:G
     OSUB 5700:GOTO 5720                      [29F2]
5540 IF bn1$="OKG" THEN h8=1003:h6=478:G
     OSUB 5700:GOTO 5720                      [FCE6]
5550 IF bn1$="RMS" THEN zz=782:GOSUB 576
     0:h8=966:h6=296.5:GOSUB 5700:er=297
     .5:fr=296.2:GOTO 5720                    [1544]
5560 IF bn1$="ROD" THEN h8=1133:h6=342:G
     OSUB 5700:GOTO 5720                      [94E6]
5570 IF bn1$="RKN" THEN h8=1129:h6=249:G
     OSUB 5700:GOTO 5720                      [C40A]
5580 IF bn1$="SWG" THEN zz=73:GOSUB 5760
     :h8=1267.3:h6=331.8:GOSUB 5700:er=3
     33:fr=331.7:GOTO 5720                    [AB02]
5590 IF bn1$="TGO" THEN h8=917:h6=368:GO
     SUB 5700:GOTO 5720                       [19B8]
5600 IF bn1$="TGL" THEN zz=121:GOSUB 576
     0:h8=1153.8:h6=485.2:GOSUB 5700:er=
     486.1:fr=484.3:GOTO 5720                 [A514]
5610 IF bn1$="TAU" THEN h8=1015:h6=314:G
     OSUB 5700:GOTO 5720                      [C1E4]
5620 IF bn1$="TRA" THEN h8=862:h6=341:GO
     SUB 5700:GOTO 5720                       [3492]
5630 IF bn1$="WSR" THEN h8=1201:h6=318:G
     OSUB 5700:GOTO 5720                      [030E]
5640 IF bn1$="WTM" THEN h8=1213:h6=276:G
     OSUB 5700:GOTO 5720                      [8A14]
5650 IF bn1$="WRB" THEN h8=1090:h6=341:G
     OSUB 5700:GOTO 5720                      [1DF4]
5660 IF bn1$="WUR" THEN h8=983:h6=386:GO
     SUB 5700:GOTO 5720                       [7DE2]
5670 IF bn1$="WYP" THEN h8=1063:h6=275:G
     OSUB 5700:GOTO 5720                      [C72E]
5680 IF bn1$="WLD" THEN h8=915:h6=443:GO
     SUB 5700:GOTO 5720                       [68A2]
5690 GOTO 5000                                [6524]
5700 IF a4=0 THEN RETURN                      [51D8]
5710 a1=h5-h6:a2=h7-h8:RETURN                 [D968]
5720 gr=h8+0.05:hr=h8-0.05:bn$=bn1$           [F006]
5730 IF a4=0 THEN 1300                        [86A6]
5740 LOCATE wff,sff:PEN 1:PRINT bn$           [9A7E]
5750 GOTO 2000                                [9E18]
5760 IF b8>zz+200 THEN 16=zz:RETURN           [2F0C]
5770 IF zx=0 THEN zx=1:16=zz:RETURN           [C0EE]
5780 PEN 1:LOCATE wff,sff:PRINT bn$:GOTO
      2000                                    [7030]
6000 LOCATE wmot,smot:PEN 3:PRINT STRING
     $(3,CHR$(231))                           [1DBC]
6010 l=0:FOR i=956 TO 716 STEP -1:SOUND
     1,i,5,1:SOUND 4,284,5,2,,,5:l=l+0.0
     29:NEXT                                  [971A]
6020 LOCATE wmot,smot:PEN 2:PRINT STRING
     $(3,CHR$(231))                           [EEBE]
6030 SOUND 1,716,500,1:SOUND 4,284,500,1
     g,,,5                                    [6B80]
6040 FOR j=1 TO 2000:NEXT                     [89CE]
6050 LOCATE wmot+1,smot:PEN 1:PRINT STRI
     NG$(2,CHR$(231))                         [F478]
6060 FOR i=716 TO 568 STEP -1:SOUND 1,i,
     5,1:SOUND 4,284,5,lg,,,5:NEXT            [7358]
6070 LOCATE wmot,smot:PEN 1:PRINT CHR$(2
     31)                                      [9570]
6080 SOUND 1,568,100,1:SOUND 4,284,100,1
     g,,,5                                    [9384]
6085 asou=568                                 [3016]
6090 RETURN                                   [939E]
6200 SYMBOL AFTER 32                          [17AA]
6210 SYMBOL 94,60,102,108,102,102,230,10
     8,96                                     [9802]
6220 SYMBOL 58,0,66,60,102,102,102,60,0       [F9FC]
6230 SYMBOL 59,36,0,102,102,102,102,62,0
                                              [9A58]
6240 SYMBOL 64,72,0,120,12,124,140,118,0
                                              [AF62]
6250 SYMBOL 42,186,108,198,198,198,108,5
     6,0                                      [ADF6]
6260 SYMBOL 43,40,198,198,198,198,198,12
     4,0                                      [8D00]
6270 SYMBOL 124,90,60,102,102,126,102,10
     2,0                                      [DC80]
6275 SYMBOL 255,0,120,204,204,120,0,0,0       [CEF0]
6280 RETURN                                   [97A0]
6300 sou=568-(d1*30-30):IF eo=1 THEN sou
     =sou+148                                 [A33A]
6302 IF n7=0 THEN sou=956                     [DCA6]
6304 IF sou<asou THEN FOR i= asou TO sou
      STEP -10:SOUND 1,i,4,1,,4:SOUND 4,
     284,4,lg,,,5:NEXT                        [5C7C]
6306 IF sou>asou THEN FOR i= asou TO sou
      STEP 10:SOUND 1,i,4,1,,4:SOUND 4,2
     84,4,lg,,,5:NEXT                         [192A]
6308 IF sou=asou THEN SOUND 1,sou,50,1,,
     4:SOUND 4,284,50,lg,,,5                  [892E]
6310 asou=sou                                 [246C]
6320 RETURN                                   [CA96]
6400 b3=CINT(b3):b5=CINT(b5):IF b3>40 TH
     EN b3=40                                 [A472]
6410 IF b3<0 THEN b3=0                        [1218]
6415 IF b5<-12 THEN b5=-12                    [B9AA]
6417 IF b5>12 THEN b5=12                      [7BFE]
6420 e4=b5:e5=-b5:bk1=2*(b3+e5):bk2=2*(b
     3+e4)                                    [594C]
6430 IF abk1=bk1 AND abk2=bk2 THEN RETUR
     N                                        [5F68]
6432 IF bk1>80 THEN bk1=80                    [01A8]
6434 IF bk2>80 THEN bk2=80                    [EDB0]
6436 IF bk1<0 THEN bk1=0                      [C5CC]
6438 IF bk2<0 THEN bk2=0                      [ECD4]
6440 IF abk1>bk1 THEN fa=0 :MOVE 290,abk
     1+31:abk1=abk1-1 ELSE fa=1:MOVE 290
     ,abk1+31:abk1=abk1+1                     [6C92]
6450 DRAWR 11,0,fa                            [9180]
6460 IF abk2>bk2 THEN fa=0 :MOVE 322,abk
     2+31:abk2=abk2-1 ELSE fa=1:MOVE 322
     ,abk2+31:abk2=abk2+1                     [4996]
6470 DRAWR 11,0,fa                            [AF84]
6480 GOTO 6430                                [C730]
6600 IF f3=1 AND sou<>956 THEN ON SQ(1)
     GOSUB 6300                               [700A]
6610 IF b6<>0 AND abh=1 THEN RETURN           [CFB2]
6620 IF b6<>0 AND abh=0 THEN LOCATE 1,1:
     PRINT lb$(4):abh=1:RETURN                [8A54]
6625 IF (h7-hr OR gr-h7 OR er-h5 OR h5-f
     r)<0 THEN RETURN                         [BABE]
6630 abh=0                                    [930E]
6640 IF b7<5 AND xc=1 THEN RETURN             [42A8]
6650 PEN 1:IF xc=0 THEN xc=1:GOTO 6720        [38E0]
6660 lb=lb+1                                  [1C4E]
6670 IF lb=4 THEN 6720                        [4E3E]
6680 IF b7>150 THEN 6720                      [3C9C]
6690 IF b7>50 AND lb=3 THEN 6720              [4ADE]
6700 IF b7>100 AND lb=2 THEN 6720             [B424]
6710 RETURN                                   [C09C]
6720 lb=0                                     [B354]
6730 lb1=lb1+1                                [880E]
6740 IF lb1=4 THEN lb1=1                      [B3D8]
6750 LOCATE 1,1                               [54B0]
6760 PRINT lb$(lb1)                           [20C4]
6770 RETURN                                   [BCA8]
```

Listing. Fliegen ohne Pilotenschein (Schluß)

Chopper



Haben Sie ein Faible für Helikopter? – Und lieben Sie Abenteuer? Dann nichts wie ran an die Buletten. Ein exzellentes Action-Spiel wartet auf Sie, das dem Hit auf dem C 64 »Fort-Apokalypse« nachempfunden wurde.

Um ein Maximum an Spielgeschwindigkeit zu erreichen, sind alle zeitkritischen Routinen in Maschinen-Sprache geschrieben. Auch die -zig verschiedenen Bilder des Spieles sind als Binär-Felder gespeichert. Das Maschinen-Programm enthält einige Routinen, die mit RSX-Befehlen aufgerufen werden. Am besten geben Sie zunächst den Basic-Teil des Spieles (Listing 1) ein und speichern ihn, denn nach dem Starten lädt er den Maschinen-Code automatisch nach. Danach geben Sie den Basic-Lader aus Listing 2 ein und speichern ihn sicherheitshalber ebenfalls. Normalerweise benötigen Sie ihn nicht mehr. Sollten sich aber Fehler eingeschlichen haben, könnte die ganze Arbeit nach einem »RUN« zunichte sein. Wenn Sie den Lader nun starten, speichert dieser selbsttätig den Maschinen-Code als Binärdatei »Chopper.bin« auf Kassette oder Diskette ab.

Der Sinn des Spieles ist schnell erklärt. Mit einem modernen Hubschrauber sollen Sie alle in einem unterirdischen Labyrinth gefangengehaltenen Menschen herausholen, um sie aus der Knechtschaft eines wirren Diktators zu befreien. Die Zeit aber drängt: Ihre Treibstoffanzeige nähert sich nur allzu schnell der Null-Marke. Doch damit nicht genug, natürlich macht Ihnen der teuflische Herrscher die Aufgabe nicht gerade leicht. Da stellen sich Ihnen bewegliche Energie-Blöcke und wachsame Kreaturen in den Weg, und die engen Passagen der Höhle verlangen Ihre volle Konzentration.

Zum Schluß noch ein heißer Tip: Die wandernden Blöcke wechseln unter Beschuß ihre Bewegungsrichtung.

(Rüdiger Möller/ja)

```
1 '(c) by ruediger moeller                        [E25C]
2 'werner sombartstr. 2  7750 konstanz            [3450]
3 'tel. 07531/54151                               [7478]
4 '                                               [ACF6]
5 'listing 1                                      [6F8E]
6 '>chopper.basic<                                [B730]
7 '                                               [CCFC]
10 MEMORY 29999:LOAD"CHOPPER.BIN":CALL 4
   0000                                           [2060]
20 MODE 0:DEFINT a-z                              [E826]
30 'init                                          [46BC]
40 PRINT CHR$(23)+"1"                             [0110]
50 DIM raum(26,6):DIM ra$(26):DEFREAL a-
   z:DIM ton(40):DEFINT a-z                       [9EBC]
60 DEG:power=300:raum=1:pilot=7:x=10:RES
   TORE 1180:y=16:maenne=0                        [CE94]
70 sink=80:alx=x:aly=y:POKE &3C,x:POKE &
   3D,y:POKE 39999,1                              [FCB8]
80 RESTORE 1180:FOR n=1 TO 25:ra$(n)="":
   FOR n1=0 TO 5:READ raum(n,n1):NEXT:NE
   XT                                             [D3F4]
90 DEFREAL a-z:GOSUB 1430:DEFINT a-z              [46C4]
100 ENV 1,5,3,1,5,-3,1:SOUND 4,0,-32760,
    0,1,0,1:ENV 2,5,3,1,15,-1,10:ENV 3,5
    ,3,1,15,-1,5                                  [ACBA]
110 CLS:fefl=0:GOSUB 570:|SWITCH,10,60,0
    :GOSUB 510:GOSUB 440:GOSUB 370:|SWIT
    CH,PEEK(39999),x,y:mflag=0                    [21BE]
120 ON raum(raum,4)+1 GOSUB 610,620,730,
    850,920,980,1050,1050,1110                    [C3F0]
130 CALL 39500:IF fefl=1 THEN |MOVE,nr,f
    x,fy                                          [7150]
140 x=PEEK(&3C):y=PEEK(&3D):fire=PEEK(&3
    E)                                            [E760]
150 siza=siza+1:IF siza>=sink THEN siza=
    0:power=power-1:GOSUB 550:GOSUB 590:
    sink=sink-3:IF sink<10 THEN sink=10           [6EF2]
160 IF x<1 OR x>74 OR y<8 OR y>46 THEN 2
    00                                            [1E3C]
170 |COLL,PEEK(39999):IF PEEK(62)=255 TH
    EN 260                                        [837C]
180 IF fire<>0 THEN GOSUB 320                     [3AB8]
190 GOTO 120                                      [F04C]
200 ' hubi aus bildschirm                         [2496]
210 IF x<5 THEN raum=raum(raum,2):x=74            [403E]
220 IF x>74 THEN raum=raum(raum,3):x=1            [A73E]
230 IF y<8 THEN raum=raum(raum,0):y=46            [DC46]
240 IF y>46 THEN raum=raum(raum,1):y=8            [F94E]
250 alx=x:aly=y:POKE &3C,x:POKE &3D,y:GO
    TO 110                                        [1E40]
260 ' hubi hat collidiert                         [069E]
270 |COLL,9:coll=0:coll=PEEK(62):|COLL,1
    0:coll=coll+PEEK(62)                          [76AA]
280 IF coll=255 THEN 470                          [41EC]
290 IF raum(raum,5)=0 THEN 470                    [C8FA]
300 IF mflag=0 THEN |SWITCH,9 ELSE |SWIT
    CH,10                                         [0C78]
310 raum(raum,5)=0:|SWITCH,11:maenne=mae
    nne+1:GOSUB 510:SOUND 1,100,0,0,2:GO
    TO 120                                        [2316]
320 ' schuss !!                                   [9BEE]
330 xk=x*8:yk=399-y*8:IF PEEK(39999)=1 T
    HEN MOVE xk+34,yk-22:DRAWR 100,-20 E
    LSE MOVE xk-2,yk-22:DRAWR -100,-20            [B228]
340 SOUND 129,0,0,0,2,0,1:ON raum(raum,4
    ) GOSUB 700,820,890,960,1020,1080,10
    80,1160                                       [A476]
350 IF PEEK(39999)=1 THEN DRAWR -100,20
    ELSE DRAWR 100,20                             [1ECA]
360 RETURN                                        [CC32]
370 ' screen aus datas lesen                      [C29C]
380 GOSUB 510                                     [5BE2]
390 CALL 39000,31900+raum*100                     [A470]
400 IF raum(raum,5)=1 THEN |INIT,10,3117
    0,8,16,PEEK(39480),PEEK(39481):|INIT
    ,9,31040,8,16,PEEK(39480),PEEK(39481
    ):|SWITCH,9:mflag=0:|INIT,11,31300,8
    ,16,PEEK(39480),PEEK(39481)                   [409C]
410 IF raum(raum,5)=0 THEN |INIT,10,3117
    0,8,16,PEEK(39480),PEEK(39481):|INIT
    ,11,31300,8,16,PEEK(39480),PEEK(3948
    1):|SWITCH,11                                 [9E14]
420 ON raum(raum,4) GOSUB 720,840,910,97
    0,1040,1090,1100,1170                         [AB16]
430 RETURN                                        [CF2E]
440 'hubis anzeigen                               [A6D6]
450 FOR n=1 TO 2:LOCATE 1,n:PRINT SPACE$
    (13):NEXT                                     [0766]
460 FOR n=0 TO pilot-1:|SWITCH,0,n*6,0:N
    EXT:RETURN                                    [E0A4]
470 ' EXPLOSION !!!                               [348C]
480 |SWITCH,PEEK(39999):SOUND 129,0,0,0,
    3,0,1:|SWITCH,12,x,y:FOR n=1 TO 1000
    :NEXT:|SWITCH,12:x=alx:y=aly:pilot=p
    ilot-1:GOSUB 440:IF pilot=0 THEN CAL
    L 38900:GOTO 60                               [458C]
490 |SWITCH,PEEK(39999),x,y:sink=80:POKE
     &3C,x:POKE &3D,y:GOTO 120                    [F0B6]
500 ' maennchen anzeigen                          [6C34]
510 LOCATE 16,3:PRINT maenne;:IF maenne<
    >25 THEN RETURN                               [FFEE]
520 CALL 38900:PRINT CHR$(7):FOR n=1 TO
    200:SOUND 1,RND*478,2,7:NEXT                  [B748]
530 MODE 1:a$="VLH#KDEHQ#DOOH#PDHQQFKHQ#
    EHIUHLW#$$#]XU#EHORKQXQJ#VWXHU]H#LFK
    #EHVRQGHUV#VFKRHQ###DE#$":FOR n=1 TO
     LEN(a$):PRINT CHR$(ASC(MID$(a$,n,1)
    )-3);:NEXT                                    [DB42]
540 CALL &BC08:END                                [0032]
550 IF raum(raum,5)=1 THEN |ANIMATE,9,10
    :mflag=mflag XOR 1                            [297C]
560 RETURN                                        [AA36]
570 'power anzeigen                               [DA02]
580 LOCATE 1,3:PRINT"POWER";                      [AAF0]
590 LOCATE 6,3:PRINT power:IF power>0 TH
    EN RETURN                                     [B124]
600 CALL 38900:GOTO 60                            [14DE]
610 CALL &BD19:CALL &BD19:RETURN                  [7C5E]
620 ' feind 1 wanderblock nach rechts             [0ACA]
630 CALL &BD19                                    [C416]
640 IF siza\8<>siza/8 THEN CALL &BD19:RE
    TURN                                          [494E]
650 IF fr=-1 THEN 680                             [7ED4]
660 fx=fx+8:IF fx-fstx>16 THEN fx=fstx            [C68A]
670 |MOVE,17,fx,fy:RETURN                         [0766]
680 fx=fx-8:IF fx<fstx THEN fx=fstx+16            [AE8A]
690 |MOVE,17,fx,fy:RETURN                         [AD6A]
700 |COLL,17:IF PEEK(&3E)=255 THEN fr=fr
    *-1                                           [6D56]
710 RETURN                                        [CE30]
720 fstx=PEEK(39482):fsty=PEEK(39483):|S
    WITCH,17,fstx,fsty:fx=fstx:fy=fsty:f
    efl=0:fr=1:ver=0:RETURN                       [1C00]
730 ' feind 1 wanderblock vertikal                [130C]
740 CALL &BD19                                    [831A]
750 IF siza\8<>siza/8 THEN CALL &BD19:RE
    TURN                                          [0052]
```

Listing 1. »Chopper« Basic-Teil

```
760 IF fr=-1 THEN 800                          [F9CC]
770 CALL &BD19                                 [5D20]
780 fy=fy+4:IF fy-fsty>12 THEN fy=fsty         [AE8C]
790 |MOVE,17,fx,fy:RETURN                      [136C]
800 fy=fy-4:IF fy<fsty THEN fy=fsty+12         [AB7A]
810 |MOVE,17,fx,fy:RETURN                      [BD5E]
820 |COLL,17:IF PEEK(&3E)=255 THEN fr=fr
    *-1                                        [0F5C]
830 RETURN                                     [4336]
840 fstx=PEEK(39482):fsty=PEEK(39483):|S
    WITCH,17,fstx,fsty:fx=fstx:fy=fsty:f
    efl=0:fr=1:ver=0:RETURN                    [7E06]
850 ' wartminen                                [9DB2]
860 CALL &BD19:CALL &BD19:|MOVE,19,fx,fy
                                               [8C68]
870 IF ABS(fx-x)>12 OR ABS(fy-y)>8 THEN
    RETURN                                     [1C16]
880 |MOVE,19,x+2,y+2:RETURN                    [134C]
890 |COLL,19:IF PEEK(&3E)=255 THEN |SWIT
    CH,19:raum(raum,4)=0                       [EBD0]
900 RETURN                                     [4232]
910 fefl=0:fx=PEEK(39482)+7:fy=PEEK(3948
    3)+3:|SWITCH,19,fx,fy:nr=19:RETURN         [A17C]
920 ' 4 block                                  [46C2]
930 blza=blza+1:IF blza>1000 THEN blza=1
                                               [247C]
940 FOR n=0 TO 3:IF blza/za(n)=blza\za(n
    ) THEN |SWITCH,17,fx+14*n,fy               [17F4]
950 NEXT:RETURN                                [722E]
960 RETURN                                     [7E3E]
970 blza=1:fx=PEEK(39482):fy=PEEK(39483)
    :fefl=0:FOR n=0 TO 3:za(n)=RND*20+14
    :NEXT:RETURN                               [533A]
980 ' bombe                                    [C31A]
990 CALL &BD19:CALL &BD19                      [B540]
1000 boza=boza+1:IF boza=100 THEN |SWITC
     H,20:|SWITCH,12,fx,fy:SOUND 129,0,0
     ,0,2,0,1:FOR n=1 TO 1000:NEXT:|SWIT
     CH,12:|SWITCH,20:GOTO 470                 [C6D6]
1010 RETURN                                    [8784]
1020 |COLL,20:IF PEEK(&3E)=0 THEN RETURN
                                               [61A0]
1030 |SWITCH,20:|SWITCH,12,fx,fy:SOUND 1
     29,0,0,0,2,0,1:FOR n=1 TO 1000:NEXT
     :raum(raum,4)=0:|SWITCH,12:boza=0:R
     ETURN                                     [D92C]
1040 boza=0:fx=PEEK(39482):fy=PEEK(39483
     ):fefl=0:|SWITCH,20,fx,fy:RETURN          [E566]
1050 ' kamikaze                                [E0F4]
1060 fx=fx+fr:IF fx>80 OR fx<1 THEN fx=f
     stx:fy=y                                  [DA34]
1070 CALL &BD19:RETURN                         [38A8]
1080 RETURN                                    [8992]
1090 treff=0:fr=-1:fefl=1:fstx=PEEK(3948
     2):fsty=PEEK(39483):fx=fstx:fy=fsty
     :nr=15:|SWITCH,15,fx,fy:RETURN            [1B2C]
1100 treff=0:fstx=PEEK(39482):fsty=PEEK(
     39483):fx=fstx:fy=fsty:nr=16:|SWITC
     H,16,fx,fy:fr=1:fefl=1:RETURN             [56C6]
1110 ' strahlen                                [0116]
1120 CALL &BD19:stza=stza+1:IF stza<50 T
     HEN CALL &BD19:RETURN                     [D268]
1130 stza=0                                    [0D28]
1140 MOVE sxk,syk:DRAW 639,0:MOVE sxk,sy
     k:DRAW 639,328:MOVE sxk,syk:DRAW 0,
     328:MOVE sxk,syk:DRAW 0,0:MOVE skx,
     syk:DRAW 0,sky:MOVE skx,syk:DRAW sk
     x,0:MOVE skx,syk:DRAW skx,328:MOVE
     skx,syk:DRAW 639,syk:|COLL,PEEK(399
     99)                                       [98D8]
1150 IF PEEK(&3E)=255 THEN x=10:y=16:rau
     m=1:POKE &3C,x:POKE &3D,y:alx=x:aly
     =y:GOTO 110                               [79BE]
1160 RETURN                                    [C690]
1170 fefl=0:sxk=PEEK(39482)*8:syk=399-PE
     EK(39483)*8:stza=0:RETURN                 [D950]
1180 DATA 0,0,0,2,1,1                          [C648]
1190 DATA 0,0,1,3,4,1                          [7454]
1200 DATA 0,8,2,4,2,1                          [1154]
1210 DATA 0,0,3,5,3,1                          [AB4C]
1220 DATA 0,10,4,0,5,1                         [82AC]
1230 DATA 0,11,0,7,2,1                         [89B0]
1240 DATA 0,12,6,8,7,1                         [98CC]
1250 DATA 3,13,7,9,6,1                         [42D8]
1260 DATA 0,14,8,0,3,1                         [36C0]
1270 DATA 5,15,0,0,1,1                         [54BA]
1280 DATA 6,16,0,12,2,1                        [7428]
1290 DATA 7,0,11,0,2,1                         [10BC]
1300 DATA 8,18,0,14,1,1                        [7924]
1310 DATA 9,0,13,0,8,1                         [4AC2]
1320 DATA 10,20,0,0,3,1                        [7C06]
1330 DATA 11,0,0,17,4,1                        [7818]
1340 DATA 0,22,16,18,6,1                       [DA92]
1350 DATA 13,23,17,19,7,1                      [C704]
1360 DATA 0,24,18,0,5,1                        [C52A]
1370 DATA 15,25,0,0,3,1                        [B924]
1380 DATA 16,0,0,22,4,1                        [7624]
1390 DATA 17,0,21,23,1,1                       [CC8A]
1400 DATA 18,0,22,0,3,1                        [7518]
1410 DATA 19,0,0,25,1,1                        [8C1E]
1420 DATA 20,0,24,0,7,1                        [A71A]
1430 MODE 0:CALL 39000,34700:PRINT CHR$(
     7):FOR n=1 TO 20:i$=INKEY$:NEXT           [61D8]
1440 LOCATE 7,15:PRINT"CHOPPER!"               [788E]
1450 LOCATE 10,16:PRINT"BY"                    [FDB8]
1460 LOCATE 3,17:PRINT"RUEDIGER MOELLER"
                                               [FF38]
1470 LOCATE 1,2:PRINT"PRESS{4 SPACE}ANY{
     4 SPACE}KEY"                              [4E78]
1480 |SWITCH,0,66,12:|SWITCH,1,10,12           [197E]
1490 |SWITCH,0,66,41:|SWITCH,1,10,41           [F088]
1500 |SWITCH,1,14,27:|SWITCH,0,60,27:|SW
     ITCH,0                                    [094C]
1510 RESTORE 1430                              [06E6]
1520 FOR n=0 TO 10:READ a:INK n,a:NEXT         [0F2A]
1530 DATA 0,25,2,3,26,23,16,16,11,13,18        [7742]
1540 RESTORE 1560                              [3FF4]
1550 DEFREAL a-z                               [480C]
1560 ENV 5,3,5,1,15,-1,3:ENV 6,5,3,1,15,
     -1,5                                      [8FAA]
1570 sp=8:RESTORE 1560                         [5D1E]
1580 FOR n=1 TO 15:READ ton(n):NEXT            [4A32]
1590 start=20:adr3=35000:adr2=35050:adr1
     =35200:adr4=35020                         [554E]
1600 IF (SQ(4) AND &X111)=4 THEN GOSUB 1
     650                                       [B200]
1610 IF (SQ(1) AND &X111)=4 THEN GOSUB 1
     690                                       [1004]
1620 IF (SQ(2) AND &X111)=4 THEN GOSUB 1
     710                                       [BFFA]
1630 IF INKEY$<>"" THEN CALL 38900:RETUR
     N                                         [A488]
1640 GOTO 1600                                 [BA16]
1650 SOUND 4,ton(start+PEEK(adr3)),2*sp,
     0,5,0,PEEK(adr3+1):adr3=adr3+2:IF a
     dr3>35015 THEN adr3=35000:|ANIMATE,
     0,1:GOSUB 1670                            [862E]
1660 RETURN                                    [BF9A]
1670 start=PEEK(adr4):adr4=adr4+1:IF sta
     rt=255 THEN adr4=35019:GOTO 1670          [749A]
1680 RETURN                                    [8B9E]
1690 SOUND 1,PEEK(adr1)+256*PEEK(adr1+1)
     ,PEEK(adr1+2)*sp,0,6:adr1=adr1+3:IF
      PEEK(adr1)=255 THEN adr1=35200           [F50A]
1700 RETURN                                    [CC90]
1710 SOUND 2,PEEK(adr2)+256*PEEK(adr2+1)
     ,PEEK(adr2+2)*sp,0,6:adr2=adr2+3:IF
      PEEK(adr2)=255 THEN adr2=35050           [3112]
1720 RETURN                                    [B094]
1730 DATA 965,851,758,716,638,568,506,47
     8,426,379,358,319,284,253,239             [4DCA]
1740 SPEED WRITE 1:SAVE"!CHOPPER.MAIN":S
     AVE"!CHOPPER.BIN",b,30000,12000           [3B7A]
```

Listing 1. »Chopper« Basic-Teil (Schluß)

```
1 '(c) by ruediger moeller                     [E25C]
2 '                                            [ACF2]
3 'listing 2                                   [7F8C]
4 '>chopper.data<                              [AD5C]
5 '                                            [ACF8]
10 DATA 30000                                  [9E9C]
20 DATA 192,192,128,0,0,128,0,0,0,192,0,
   64,64,200,128,64,196,200                    [98DE]
30 DATA 192,192,196,200,192,192,196,192,
   192,64,64,192,128,0,0,192                   [58A4]
40 DATA 0,0,64,0,128,0,128,0,64,0,192,19
   2,192,0,0,0,0,0,0,0,0,0                     [F110]
50 DATA 0,0,0,0,0,64,192,192,0,0,64,0,12
   8,0,192,0,128,64,196,128                    [04CC]
60 DATA 192,192,196,200,192,192,196,200,
   128,192,192,200,0,64,192                    [AA2E]
70 DATA 128,0,0,192,0,0,64,0,128,0,128,0
   ,64,0,192,192,192,0,0,0                     [C054]
80 DATA 0,0,0,0,0,0,0,0,0,195,195,0,195,
   195,67,131,131,0,65,2,10                    [C7A6]
90 DATA 10,5,15,11,1,5,5,2,5,0,5,65,7,3,
   7,11,131,195,75,135,195                     [0C7C]
100 DATA 1,65,65,5,10,0,65,1,10,15,195,1
    95,15,5,195,195,195,2,65                   [7B0C]
110 DATA 0,11,131,135,195,130,67,130,7,1
    95,7,65,0,11,0,65,3,65,195                 [57B4]
120 DATA 0,130,15,10,3,1,5,65,1,2,10,0,3
    ,0,67,15,130,1,11,0,2,1                    [17F8]
130 DATA 65,65,0,1,1,0,2,0,0,2,0,0,2,0,2
    ,0,0,2,0,0,2,0,2,0,0,0,0                   [26F8]
140 DATA 0,0,0,2,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                     [1DAA]
150 DATA 0,0,0,0,0,0,2,10,0,0,0,2,0,0,10
    ,0,2,0,10,135,195,11,11                    [F6BA]
160 DATA 5,11,15,2,10,2,1,5,15,10,2,5,15
    ,5,75,135,0,75,195,67,2                    [9962]
170 DATA 75,130,3,1,7,2,75,65,135,65,135
    ,1,135,10,11,10,135,15,131                 [869A]
180 DATA 135,131,5,2,67,2,0,10,130,130,2
    ,75,0,15,5,65,75,3,3,5,0                   [5CB6]
190 DATA 65,2,5,0,130,195,67,75,67,7,3,6
    5,130,195,195,131,3,1,0                    [7AAE]
200 DATA 1,1,67,2,195,130,3,0,5,0,0,0,0,
    0,0,0,1,0,0,0,0,0,0,0,65                   [C428]
```

Listing 2. »Chopper« Basic-Lader

Listing 2. »Chopper« Basic-Lader (Fortsetzung)

```
210 DATA 0,0,0,0,0,0,0,2,0,0,0,0,0,0,3,6
    7,0,0,0,0,0,0,1,65,0,0,0              [6EE6]
220 DATA 0,1,0,75,15,0,0,0,0,0,7,130,65,
    0,0,0,0,0,3,3,130,0,0,0               [3DCA]
230 DATA 0,7,130,2,195,0,0,0,0,67,1,65,5
    ,0,0,0,3,130,131,67,195               [8842]
240 DATA 0,0,1,67,65,65,195,195,0,0,0,13
    1,10,195,130,2,0,1,67,7               [B16C]
250 DATA 65,75,11,65,0,2,5,0,2,5,15,1,3,
    65,131,131,195,195,75,67              [48FA]
260 DATA 0,0,2,0,0,0,0,0,0,0,1,2,2,0,0,0
    ,0,0,0,5,1,0,0,0,0,0,7,2              [CED8]
270 DATA 0,0,0,0,0,0,65,131,131,195,0,0,
    0,0,3,5,3,0,0,0,0,0,131               [38DE]
280 DATA 2,67,130,0,0,0,0,65,65,65,3,5,1
    ,0,0,130,3,7,0,10,11,2,0              [7878]
290 DATA 135,5,15,15,0,10,0,0,131,195,10
    ,7,1,7,0,2,195,0,7,11,11              [8EA0]
300 DATA 3,11,0,195,135,135,2,7,7,5,11,6
    7,130,130,195,75,75,130               [588C]
310 DATA 131,0,10,65,65,131,195,195,7,2,
    11,15,1,2,67,195,131,2,11             [223C]
320 DATA 3,7,65,75,15,5,7,3,0,2,75,0,1,1
    0,0,1,0,2,65,2,130,67,135             [9E0E]
330 DATA 7,0,1,65,195,135,135,3,3,0,0,0,
    7,65,195,130,15,0,0,0,1               [8F4C]
340 DATA 1,7,65,135,0,0,0,0,2,130,5,67,0
    ,0,0,65,1,5,2,65,0,0,0,65             [93E4]
350 DATA 67,131,3,75,0,0,0,0,0,1,7,75,0,
    0,0,0,0,0,2,67,0,0,0,0,0              [824C]
360 DATA 0,0,75,0,0,0,0,0,0,0,67,0,0,0,0
    ,0,0,0,2,0,0,0,0,0,0,0,5              [2AF6]
370 DATA 0,0,0,0,0,0,0,1,0,11,0,11,11,5,
    5,11,10,0,10,1,3,3,135,3              [4620]
380 DATA 11,2,15,15,0,67,67,195,65,195,1
    35,0,2,1,195,10,2,11,5,0              [84E2]
390 DATA 7,135,135,1,15,2,10,0,3,130,1,3
    ,11,15,0,0,3,195,65,135               [FE4C]
400 DATA 2,10,0,0,195,10,1,3,5,0,0,0,195
    ,3,1,67,10,10,0,0,65,15               [0412]
410 DATA 11,2,2,0,0,0,2,0,0,130,0,10,0,0
    ,67,11,65,1,5,0,0,0,65,10             [C398]
420 DATA 10,15,0,0,0,0,130,65,0,0,0,0,0,
    0,195,10,0,0,0,0,0,0,135              [402E]
430 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
    0,1,2,2,2,3,1,3,2,2,0,0               [3778]
440 DATA 0,65,3,0,15,1,1,1,135,75,75,2,0
    ,5,3,15,1,0,131,5,1,195               [2342]
450 DATA 0,135,11,2,130,130,67,75,7,1,65
    ,0,11,135,130,67,131,3,65             [E236]
460 DATA 7,131,7,10,75,135,65,130,1,135,
    130,130,1,130,15,5,130,0              [33BC]
470 DATA 0,195,11,130,135,11,130,131,130
    ,195,2,7,131,3,5,75,1,130             [2628]
480 DATA 3,10,7,10,7,130,5,1,10,2,5,131,
    5,67,67,130,11,5,11,65,65             [1C24]
490 DATA 65,7,0,3,0,0,10,10,2,5,1,10,1,0
    ,2,3,0,5,11,3,64,8,0,0,0              [4652]
500 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [85A6]
510 DATA 0,0,0,0,0,0,8,0,0,128,4,4,4,8,8
    ,4,8,168,4,12,4,0,8,4,4               [EE2A]
520 DATA 168,4,4,4,8,8,4,8,160,0,4,4,0,8
    ,4,0,240,0,0,4,8,12,0,0               [8538]
530 DATA 160,0,0,0,0,0,0,0,160,0,0,0,0,0
    ,0,0,240,0,0,0,0,0,0,0,160            [07BA]
540 DATA 160,0,0,0,0,0,0,48,48,48,48,48,
    48,48,48,48,48,48,48,48               [73D4]
550 DATA 48,48,48,64,8,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [F02C]
560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,8,4,0
    ,8,4,0,128,0,4,4,0,8,8,0              [5230]
570 DATA 168,0,0,0,0,0,0,0,168,0,0,0,0,0
    ,0,0,160,0,4,0,0,0,8,0,240            [29FA]
580 DATA 0,8,4,0,8,4,0,160,0,0,4,0,8,0,0
    ,160,0,0,0,0,0,0,0,240,0              [143E]
590 DATA 0,0,0,0,0,0,160,160,0,0,0,0,0,0
    ,48,48,48,48,48,48,48,48              [C2C4]
600 DATA 48,48,48,48,48,48,48,48,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [8588]
610 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [B0AA]
620 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [0CAC]
630 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [80AE]
640 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,48,
    48,48,48,48,48,48,48,48               [1250]
650 DATA 48,48,48,48,48,48,48,4,64,0,0,0
    ,0,0,0,0,16,32,0,204,0,0              [A6CA]
660 DATA 0,0,0,0,32,0,0,0,0,0,32,0,68,0,
    68,136,0,16,0,0,68,32,204             [B9DC]
670 DATA 204,152,0,0,0,68,152,136,204,68
    ,136,0,0,68,68,32,136,136             [8E68]
680 DATA 136,0,0,0,136,64,12,68,68,0,0,0
    ,204,64,0,76,68,0,0,0,140             [322A]
690 DATA 200,0,8,0,0,68,204,68,76,0,68,1
    36,0,0,16,0,0,136,204,32              [EED4]
700 DATA 48,16,0,0,68,8,136,0,16,32,0,0,
    0,68,204,0,0,0,0,0,0,0,0              [1F78]
710 DATA 0,0,0,0,0,0,0,0,0,0,0,0,40,0,0,
    0,60,0,0,0,60,60,60,8,60              [9A18]
720 DATA 48,52,60,60,60,60,40,0,32,0,32,
    0,0,0,0,0,0,0,0,0,68,204              [F278]
730 DATA 96,100,0,0,0,0,0,0,20,0,0,0,60,
    4,60,60,60,60,56,48,60,20             [99E6]
740 DATA 60,60,60,16,0,16,0,0,0,0,0,0,0,
    0,0,0,68,16,96,48,204,0               [DA2E]
750 DATA 0,0,0,20,204,0,0,136,0,0,136,10
    4,60,204,156,204,64,0,156             [4506]
760 DATA 148,104,0,0,136,156,0,0,20,20,0
    ,0,0,204,0,0,0,64,0,0,0               [1C06]
770 DATA 0,128,136,0,0,60,68,136,0,40,15
    6,68,0,204,148,196,136,156            [BCDC]
780 DATA 156,60,68,128,0,136,0,60,68,0,0
    ,204,0,0,0,64,0,0,64,0,0              [1CB8]
790 DATA 0,0,48,48,48,48,48,48,48,48,48,
    48,48,48,48,48,48,48,48               [F03C]
800 DATA 48,48,48,48,48,48,48,48,48,48,4
    8,48,48,48,48,48,48,48,48             [CB1C]
810 DATA 48,48,48,48,48,48,48,48,48,48,4
    8,48,48,48,48,48,48,48,48             [881E]
820 DATA 48,48,48,48,48,48,48,48,48,48,4
    8,48,48,48,48,48,48,48,48             [E920]
830 DATA 48,48,48,48,48,48,48,48,48,48,4
    8,48,48,48,48,48,48,48,48             [9622]
840 DATA 48,48,48,48,48,48,48,48,48,48,4
    8,48,48,48,48,48,48,48,48             [D724]
850 DATA 48,48,48,48,48,48,48,48,48,48,4
    8,48,48,48,48,48,0,0,156              [7996]
860 DATA 108,156,108,0,0,0,0,0,0,0,0,68,
    68,0,0,0,136,0,64,192,68              [5AC0]
870 DATA 0,128,0,136,0,128,68,68,0,64,0,
    0,0,20,40,0,0,48,48,0,16              [63C8]
880 DATA 48,48,32,16,48,48,32,16,48,48,3
    2,16,48,48,32,0,48,48,0               [46E6]
890 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [12BE]
900 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [B1AE]
910 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0                [E5B0]
920 DATA 0,0,0,0,8,8,8,8,8,8,8,8,8,8,8,2
    ,8,8,8,8,8,8,2,2,8,1,6,8              [4CFC]
930 DATA 8,8,8,2,1,1,8,1,1,2,2,2,1,1,1,1
    ,8,1,1,10,1,1,1,1,1,1,8               [D5EE]
940 DATA 1,4,3,3,3,3,9,1,1,8,3,8,8,8,8,8
    ,8,3,3,8,8,8,8,8,8,8,8,8              [9B00]
950 DATA 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8
    ,8,8,8,8,8,8,8,8,8,8,8,8              [7F78]
960 DATA 8,8,8,1,1,6,8,8,8,8,8,8,8,1,1,1
    ,6,8,8,8,8,8,8,1,1,1,10               [1F8C]
970 DATA 1,1,1,1,1,1,1,1,9,4,3,3,3,3,3,3
    ,1,4,8,8,8,8,8,8,8,8,4,8              [54AC]
980 DATA 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8
    ,8,8,8,8,8,8,8,8,8,8,8,8              [637E]
990 DATA 8,8,8,8,8,8,8,8,8,8,8,8,8,8,2,2
    ,2,8,8,8,8,8,8,8,1,1,9,8              [6142]
1000 DATA 8,8,8,2,8,8,10,8,8,8,2,2,1,1,6
     ,8,1,8,8,8,1,1,3,1,1,8,1             [BBA6]
1010 DATA 8,8,7,1,1,8,1,6,8,1,8,7,1,1,4,
     8,1,1,2,1,2,1,1,4,8,8,5,1            [6FE2]
1020 DATA 1,1,1,1,4,8,8,8,8,5,1,2,2,8,8,
     8,8,8,8,8,1,1,1,1,8,8,8,8            [361E]
1030 DATA 8,8,8,8,8,8,8,8,8,1,1,1,1,1,6,
     8,8,8,8,3,3,3,5,1,1,1,1,9            [F420]
1040 DATA 8,1,6,8,8,1,6,5,1,1,1,1,1,6,8,
     1,1,6,3,3,3,5,1,1,8,1,1,1            [FDC8]
1050 DATA 8,8,8,8,1,1,8,1,1,4,8,8,8,8,1,
     2,7,1,4,8,8,8,8,8,1,10,1             [84C4]
1060 DATA 1,8,8,8,8,8,8,3,3,3,8,8,8,8,8,
     8,8,8,8,8,8,8,8,8,8,8,8,8            [809E]
1070 DATA 8,8,2,2,2,2,2,8,7,6,8,7,1,1,1,
     1,1,8,1,1,6,1,1,4,3,5,1,8            [53E0]
1080 DATA 5,1,8,1,3,8,8,8,1,8,8,1,8,1,6,
     2,8,8,1,8,8,1,8,10,1,1,8             [B6BA]
1090 DATA 8,1,8,8,1,6,2,7,1,8,8,1,8,8,1,
     1,1,1,1,8,8,9,8,8,3,3,3,5            [6C1E]
1100 DATA 1,8,8,8,8,8,8,2,8,8,8,8,8,8,8,
     8,8,10,9,8,8,8,8,2,2,8,8             [751A]
1110 DATA 1,8,2,2,2,7,1,1,8,8,1,7,1,1,1,
     1,1,1,8,8,1,1,1,3,3,7,1,1            [46B2]
1120 DATA 8,8,5,1,4,7,1,1,1,1,8,8,8,3,7,
     1,1,4,3,3,8,8,8,8,1,1,3,8            [F704]
1130 DATA 8,8,8,8,8,7,1,1,6,8,8,8,8,8,8,
     1,1,1,1,8,8,8,8,8,8,8,8,8            [4B6C]
1140 DATA 8,8,8,8,2,2,2,2,2,2,2,2,2,2,1,
     1,1,1,1,1,1,1,1,1,1,1,1,1            [F88C]
1150 DATA 1,1,1,1,1,1,1,10,1,1,1,1,1,1,1
     ,1,8,2,2,2,2,2,2,2,8,8,8             [872E]
1160 DATA 7,1,1,1,1,1,1,6,8,8,1,1,1,1,1,
     1,5,1,8,8,5,1,1,1,1,4,8,9            [B6C6]
1170 DATA 8,8,8,3,5,1,1,8,8,8,8,8,8,8,1,
     1,1,1,6,8,8,8,8,8,5,1,1,1            [5F36]
1180 DATA 1,6,8,2,8,8,8,5,4,8,1,1,8,1,6,
     2,2,2,2,7,1,1,8,1,1,1,1,1            [42E2]
1190 DATA 1,1,1,10,1,5,1,1,1,1,1,9,1,1,1
     ,8,5,1,1,6,3,8,3,3,3,8,8             [9162]
1200 DATA 5,1,1,6,8,8,8,8,8,8,8,1,1,1,6,
     8,8,8,8,8,8,8,1,1,1,6,8,8            [DA40]
1210 DATA 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
     8.8.8,8,8,8,8,8,8,8,8,8,8            [46C4]
```

```
1220 DATA 8,8,8,8,8,8,8,8,8,8,8,8,8,1,1,6,
     2,2,2,8,8,8,8,10,1,1,1,1            [95D4]
1230 DATA 1,6,8,8,8,8,5,9,1,1,1,1,8,8,8,
     8,8,5,1,1,1,1,8,8,8,8,8,8           [0A3C]
1240 DATA 5,1,1,1,8,8,8,8,8,8,8,1,1,1,1,6,
     8,8,8,8,8,8,8,1,8,8,8,8,8           [F650]
1250 DATA 8,8,8,8,1,1,6,8,8,8,8,8,8,8,8,5,
     1,1,6,8,8,8,8,8,8,8,5,1,1           [1A64]
1260 DATA 8,8,8,8,8,8,8,7,1,4,8,8,9,10,1
     ,1,1,1,8,8,8,8,8,8,8,8,8            [20FC]
1270 DATA 5,1,8,8,8,8,8,8,8,8,8,1,8,8,8,
     8,8,2,2,2,7,1,8,8,8,8,8,1           [F66C]
1280 DATA 1,1,1,1,8,8,8,2,7,1,1,1,1,8,8,
     8,8,1,1,1,1,1,1,1,6,8,8,1           [2FE0]
1290 DATA 1,4,3,3,5,1,6,2,8,1,4,8,8,8,1,
     1,1,1,8,10,8,8,8,7,1,4,8            [1CAA]
1300 DATA 1,8,1,8,2,7,1,1,8,8,1,8,1,8,1,
     1,1,4,8,8,1,8,1,8,9,3,3,8           [D102]
1310 DATA 8,8,3,8,1,8,8,8,8,8,8,8,8,8,1,
     8,8,8,8,8,8,8,8,8,8,2,7,1           [9B84]
1320 DATA 1,4,8,8,8,8,7,1,1,1,1,6,8,8,8,
     7,10,1,4,5,1,1,6,8,8,1,1            [78A4]
1330 DATA 4,8,8,5,1,1,6,8,1,1,8,8,8,8,5,
     1,1,8,1,1,8,8,8,8,8,5,9,8           [693E]
1340 DATA 1,1,8,8,8,8,8,8,8,8,8,1,1,6,2,
     2,2,2,2,8,8,5,1,1,1,1,1,1           [ACFA]
1350 DATA 1,8,8,8,3,3,3,3,3,3,8,8,8,8,8,
     8,1,1,1,4,8,8,8,2,2,7,1,4           [F322]
1360 DATA 8,8,8,8,8,10,1,1,1,8,8,8,8,8,8
     ,8,5,1,9,8,8,8,8,8,8,8,8            [B81E]
1370 DATA 1,6,2,2,2,2,2,8,8,7,1,1,1,1,1,
     1,1,8,5,1,1,1,1,1,1,1,1,8           [B4AA]
1380 DATA 8,5,1,1,1,1,1,4,8,8,8,8,5,1,1,
     1,4,8,8,8,8,8,8,1,1,1,8,8           [131E]
1390 DATA 8,8,8,8,1,1,1,1,8,8,8,8,9,2,2,
     1,2,2,2,2,8,8,1,1,1,1,1,1           [ACF6]
1400 DATA 1,1,8,8,5,4,3,3,5,10,1,6,8,2,8
     ,8,8,8,8,5,1,1,8,1,6,2,2            [BBA2]
1410 DATA 2,8,8,1,4,8,1,1,1,1,1,6,7,1,8,
     8,8,8,5,1,1,1,1,1,4,8,8,8           [39F8]
1420 DATA 8,3,3,3,3,3,8,8,8,8,8,8,8,8,8,
     8,8,8,8,8,8,1,1,1,1,1,8,8           [7852]
1430 DATA 8,8,8,1,1,1,1,4,8,8,8,8,8,1,1,
     1,1,8,8,8,8,8,8,1,1,1,1,8           [F41C]
1440 DATA 8,8,8,8,2,1,1,1,1,8,8,8,8,8,9,
     1,1,1,1,8,8,8,8,8,8,10,1            [88D0]
1450 DATA 1,1,8,8,8,8,8,8,1,1,1,1,8,8,8,
     8,8,8,1,1,1,1,8,8,8,8,8,8           [DE44]
1460 DATA 1,1,1,1,8,8,8,8,1,8,8,8,2,2,2,
     2,8,8,1,8,8,8,1,1,1,1,8,8           [E016]
1470 DATA 1,8,8,9,1,4,5,1,8,8,1,8,2,2,1,
     8,8,1,1,8,1,8,1,1,1,8,8,5           [8210]
1480 DATA 1,8,1,8,1,4,2,2,2,2,2,8,1,8,1,
     1,1,1,1,1,8,8,1,6,2,2,2,7           [DBCE]
1490 DATA 1,1,8,8,1,10,1,1,1,1,1,4,8,8,5
     ,4,3,3,3,3,3,8,8,8,8,8,8            [2FAA]
1500 DATA 8,8,8,8,8,8,8,7,1,1,1,1,1,1,6,
     8,8,1,1,1,1,1,1,1,1,8,1,1           [F8E2]
1510 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,2,9,1,
     1,10,1,8,1,1,1,1,1,1,1,1,1          [4508]
1520 DATA 1,8,1,1,1,1,1,1,1,1,8,8,1,1,1,
     1,1,1,1,1,8,8,5,1,1,1,1,1           [A492]
1530 DATA 1,4,8,8,8,8,8,1,1,1,1,8,8,8,8,8,
     8,2,1,1,1,8,8,8,8,8,8,1,1           [A52E]
1540 DATA 1,1,8,8,8,2,2,7,1,1,1,1,6,2,2,
     1,1,1,1,1,1,1,1,1,1,1,1,10          [B736]
1550 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
     1,1,1,3,3,5,1,1,1,1,1,4,3,3         [F868]
1560 DATA 8,8,6,1,1,1,1,8,8,8,8,8,9,1,1,1,
     1,1,8,8,8,8,8,8,5,1,1,1,1,8         [B516]
1570 DATA 8,8,8,8,8,2,2,2,2,2,2,8,8,8,8,
     1,1,1,1,1,1,8,8,8,8,1,4,3           [4D1A]
1580 DATA 3,5,1,8,1,8,8,1,8,2,2,2,1,8,1,
     6,7,1,8,7,1,1,1,8,1,1,1,1           [CCE6]
1590 DATA 8,1,1,4,8,8,8,2,2,2,7,10,8,8,8
     ,8,8,9,1,1,1,1,8,8,8,8,8            [9BE4]
1600 DATA 8,3,3,5,1,8,8,8,8,8,8,8,8,8,1,
     8,8,8,8,8,8,2,1,1,1,1,8,8           [D250]
1610 DATA 8,8,8,1,1,1,1,2,8,8,8,8,8,2,1,
     1,1,1,8,8,8,8,8,1,1,1,1,1,2         [6200]
1620 DATA 8,8,8,8,8,2,1,1,1,1,9,8,8,8,
     1,1,1,1,2,8,8,8,8,8,2,1,1           [6E12]
1630 DATA 1,1,8,8,8,8,8,8,1,1,1,1,2,8,8,8,
     8,8,2,1,1,1,10,8,8,8,8,8            [85C4]
1640 DATA 1,1,1,1,8,8,8,8,8,8,8,8,8,8,8,
     8,8,8,8,10,1,1,1,1,1,1,6            [B1CC]
1650 DATA 8,8,1,1,4,2,2,5,1,1,1,8,5,1,1,
     1,1,1,1,4,8,8,2,2,2,2,2,2           [C9BE]
1660 DATA 2,2,8,8,9,10,1,1,1,1,1,1,8,8,2
     ,2,2,2,2,2,5,1,8,8,1,1,1            [F270]
1670 DATA 1,1,1,1,1,1,1,8,5,1,1,1,1,1,1,4,
     8,8,8,3,3,3,3,3,3,8,8,8,8           [E1E6]
1680 DATA 8,1,1,1,1,1,8,8,8,8,8,7,1,1,1,9,
     8,8,8,1,1,1,1,1,6,8,8,2             [8814]
1690 DATA 3,3,3,3,5,1,1,6,8,1,8,8,8,8,8,
     5,1,1,1,1,8,8,8,8,8,8,5,10          [D88E]
1700 DATA 1,1,8,8,8,8,8,8,2,1,1,8,1,1,6,
     8,8,2,1,1,4,8,5,1,1,1,1,1           [A6EC]
1710 DATA 1,4,8,8,8,3,3,3,3,3,3,8,8,8,8,
     8,8,1,1,1,1,8,8,8,8,8,8,5           [B53E]
1720 DATA 1,1,6,8,8,8,8,8,8,8,5,10,1,6,8
     ,8,1,8,8,8,8,5,1,1,6,8,1            [A2E0]
1730 DATA 6,8,8,8,8,5,1,1,8,1,1,1,6,8,8,
     7,1,1,8,8,5,1,1,6,9,1,1,1           [BB12]
1740 DATA 8,8,8,5,1,1,6,7,1,1,8,8,8,8,5,
     1,1,1,1,4,8,8,8,8,8,3,3,3           [912C]
1750 DATA 3,8,8,8,8,8,8,7,1,8,8,8,8,8,8,
     8,7,1,1,8,8,8,8,8,8,7,1,1           [F180]
1760 DATA 4,8,8,2,2,8,7,1,1,4,8,8,8,1,1,
     8,5,1,4,8,8,8,7,1,1,8,8,1           [2F2E]
1770 DATA 8,9,1,10,1,1,4,8,8,1,8,8,8,8,7
     ,1,8,8,1,6,2,2,7,1,1,8              [CECE]
1780 DATA 8,5,1,1,1,1,1,1,4,8,8,8,8,8,8,
     8,8,8,8,8,8,8,8,1,1,1,1,8           [FD42]
1790 DATA 8,8,8,8,8,5,1,1,4,8,8,8,8,8,8,
     8,1,1,8,8,8,8,8,8,8,8,5,1           [3584]
1800 DATA 8,2,9,8,1,10,8,2,2,1,7,1,1,8,5
     ,1,6,1,1,1,1,1,1,8,8,1,1            [C374]
1810 DATA 1,1,1,1,1,4,8,8,5,1,1,1,1,1,1,
     2,8,8,8,5,1,1,1,1,1,1,8,8           [01C2]
1820 DATA 8,8,8,8,8,8,8,8,8,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,0,0,0,0           [B7A2]
1830 DATA  34700                         [E788]
1840 DATA 8,2,8,2,2,2,2,8,2,8,8,1,7,1,1,
     1,1,6,1,8,8,7,1,1,1,1,1,1           [3DDE]
1850 DATA 6,8,8,1,1,1,1,1,1,1,1,8,8,1,1,
     1,1,1,1,1,1,8,8,1,1,1,1,1           [8DAE]
1860 DATA 1,1,1,8,8,1,1,1,1,1,1,1,1,8,8,
     5,1,1,1,1,1,4,8,8,1,5,1             [2DBC]
1870 DATA 1,1,1,4,1,8,8,3,8,3,3,3,3,8,3,
     8,0,0,0,0,0,0,0,0,0,0,0,0           [47A0]
1880 DATA  35000                         [7D86]
1890 DATA 0,20,0,0,2,1,2,0,4,20,4,0,2,1,
     4,20,0,0,0,1,1,1,1,1,1,1            [3E12]
1900 DATA 4,4,20,20,4,1,4,1,20,20,20,5,1
     ,255,0,0,0,0,0,0,0,0                [5D2C]
1910 DATA 0,0,112,179,0,12,159,0,4,142,0
     ,8,159,0,2,179,0,6,179,0            [47D8]
1920 DATA 16,190,0,16,0,0,2,179,0,2,179,
     0,2,179,0,4,179,0,6,190,0           [7638]
1930 DATA 4,190,0,4,239,0,4,239,0,4,0,0,
     2,179,0,2,179,0,2,179,0,4           [B72C]
1940 DATA 179,0,6,190,0,4,190,0,4,239,0,
     4,239,0,4,0,0,48,0,32,255           [E5F0]
1950 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,0,0,0,0           [3E1A]
1960 DATA  35200                         [1988]
1970 DATA 0,0,64,190,0,12,159,0,4,159,0,
     8,190,0,2,213,0,4,239,0,18          [7492]
1980 DATA 142,0,12,127,0,4,119,0,8,127,0
     ,2,142,0,6,142,0,16,159,0           [7522]
1990 DATA 16,0,0,2,142,0,2,142,0,2,142,0
     ,4,142,0,6,159,0,4,159,0            [A6A4]
2000 DATA 4,190,0,4,239,0,4,0,0,2,142,0,
     2,142,0,2,142,0,4,142,0,6           [FDD0]
2010 DATA 159,0,4,159,0,4,190,0,4,239,0,
     4,190,0,12,179,0,4,159,0            [BAD0]
2020 DATA 8,142,0,2,134,0,4,127,0,18,159
     ,0,8,253,0,4,238,0,20,255           [CD24]
2030 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,0,0,0,0           [5506]
2040 DATA  38900                         [A988]
2050 DATA 6,255,33,0,192,205,2,152,16,24
     8,205,20,188,201,22,0,237           [D950]
2060 DATA 95,95,25,216,175,54,0,24,244,0
     ,0,0,0,0,0,0,0,0,0,0,0,0            [8180]
2070 DATA  39000                         [8F7E]
2080 DATA 254,1,192,175,50,213,152,62,8,
     50,214,152,221,86,1,221,94          [5ACC]
2090 DATA 0,6,100,26,254,9,204,183,152,2
     54,10,204,198,152,205,126           [3964]
2100 DATA 152,19,16,239,201,245,197,213,
     229,254,1,40,20,245,205,124         [202A]
2110 DATA 157,58,213,152,253,119,1,58,21
     4,152,253,119,0,241,205,22          [FFC6]
2120 DATA 157,58,213,152,198,8,50,213,15
     2,254,80,32,12,58,214,152           [097E]
2130 DATA 198,4,50,214,152,175,50,213,15
     2,225,209,193,241,201,58            [4314]
2140 DATA 213,152,50,56,154,58,214,152,5
     0,57,154,62,1,201,58,213            [6B06]
2150 DATA 152,50,58,154,58,214,152,50,59
     ,154,62,1,201,0,48,0,0,0            [ACE6]
2160 DATA  39500                         [C688]
2170 DATA 175,50,62,0,58,244,180,71,203,
     64,40,7,58,61,0,61,50,61            [A206]
2180 DATA 0,203,72,40,7,58,61,0,60,50,61
     ,0,203,80,40,14,58,60,0,61          [0990]
2190 DATA 50,60,0,205,170,154,175,50,63,
     156,203,88,40,15,58,60,0            [A206]
2200 DATA 60,50,60,0,205,205,154,62,1,50
     ,63,156,203,96,40,5,62,255          [F6AE]
2210 DATA 50,62,0,58,60,0,103,58,61,0,11
     1,58,63,156,205,25,189,205          [D0BE]
2220 DATA 47,157,201,58,63,156,254,0,200
     ,197,62,1,205,22,157,193            [121A]
2230 DATA 175,205,124,157,58,60,0,253,11
     9,1,58,61,0,253,119,0,62            [9306]
2240 DATA 0,205,22,157,201,58,63,156,254
     ,1,200,197,175,205,22,157           [8D6E]
2250 DATA 62,1,205,124,157,58,60,0,253,1
     19,1,58,61,0,253,119,0,62           [B55A]
2260 DATA 1,205,22,157,193,201,0,0,0,0,0
     ,0,0,0,0,0,0,0,0,0,0,0,0            [6F2A]
```



```
2270 DATA  39800                            [DB92]
2280 DATA 0,0,0,0,0,0,0,0,14,0,10,0,173,
     32,15,1,170,32,26,47,117               [B478]
2290 DATA 0,70,0,20,0,140,32,38,67,48,44
     ,38,67,48,44,38,56,48,44               [5128]
2300 DATA 38,48,48,44,38,48,48,44,38,56,
     48,44,38,48,48,44,38,48,48             [154A]
2310 DATA 44,38,48,48,44,38,67,48,44,38,
     48,48,44,38,52,48,44,38,52             [2F34]
2320 DATA 48,44,38,67,56,44,38,56,48,44,
     38,52,48,0,70,0,30,0,140               [221E]
2330 DATA 32,38,67,52,44,38,67,56,44,38,
     67,48,44,38,67,48,44,38,67             [6442]
2340 DATA 52,44,38,67,56,44,38,67,48,44,
     38,67,48,44,38,67,52,44,38             [763E]
2350 DATA 67,48,44,38,67,48,44,38,52,48,
     44,38,52,48,44,38,67,48,44             [D342]
2360 DATA 38,56,48,44,38,48,48,0,70,0,40
     ,0,140,32,38,48,48,44,38               [422A]
2370 DATA 67,48,44,38,48,48,44,38,48,48,
     44,38,52,48,44,38,48,48,44             [DE4C]
2380 DATA 38,56,48,44,38,48,48,1,1,73,15
     6,33,116,156,195,209,188               [AC62]
2390 DATA 90,156,195,120,156,195,244,156
     ,195,148,156,195,185,156               [8F7A]
2400 DATA 195,201,156,73,78,73,212,67,79
     ,76,204,83,87,73,84,67,200             [6F2A]
2410 DATA 77,79,86,197,65,78,73,77,65,84
     ,197,0,0,0,73,156,254,6,192            [06A6]
2420 DATA 221,126,10,221,94,8,221,86,9,2
     21,70,6,221,78,4,221,110               [E4F0]
2430 DATA 0,221,102,2,205,254,156,201,25
     4,1,40,26,254,3,192,221,126            [DEFC]
2440 DATA 4,205,124,157,221,126,0,253,11
     9,0,221,126,2,253,119,1,221            [0702]
2450 DATA 126,4,24,3,221,126,0,205,22,15
     7,201,254,3,192,221,126,4              [1540]
2460 DATA 221,102,2,221,110,0,205,47,157
     ,201,254,4,40,23,254,2,192             [2C94]
2470 DATA 221,126,0,205,124,157,253,102,
     1,253,110,0,221,86,2,221               [16D0]
2480 DATA 94,0,24,12,221,86,6,221,94,4,2
     21,102,2,221,110,0,205,74              [8538]
2490 DATA 157,201,254,1,192,221,126,0,20
     5,189,157,201,197,205,124              [6B7E]
2500 DATA 157,193,253,117,0,253,116,1,25
     3,112,2,253,113,3,253,115              [A550]
2510 DATA 4,253,114,5,201,205,124,157,25
     3,110,0,253,102,1,253,70               [A3D0]
2520 DATA 2,253,78,3,253,94,4,253,86,5,2
     05,96,157,201,229,205,22               [4B2E]
2530 DATA 157,225,253,117,0,253,116,1,25
     3,70,2,253,78,3,253,86,5               [8D18]
2540 DATA 253,94,4,205,96,157,201,229,21
     3,122,205,22,157,209,225               [0024]
2550 DATA 123,205,124,157,253,117,0,253,
     116,1,205,22,157,201,197               [6602]
2560 DATA 213,205,240,157,209,193,197,22
     9,26,174,119,19,44,204,230             [0602]
2570 DATA 157,16,246,225,193,13,200,205,
     214,157,24,235,253,33,21               [3814]
2580 DATA 158,254,0,200,71,253,35,253,35
     ,253,35,253,35,253,35,253              [F188]
2590 DATA 35,16,242,201,62,0,50,62,0,197
     ,213,205,240,157,209,193               [F90C]
2600 DATA 197,229,26,190,32,16,19,44,204
     ,230,157,16,245,225,193,13             [72F0]
2610 DATA 200,205,214,157,24,234,225,225
     ,62,255,50,62,0,201,205,124            [130E]
2620 DATA 157,253,110,0,253,102,1,253,70
     ,2,253,78,3,253,94,4,253               [5FFE]
2630 DATA 86,5,205,147,157,201,124,198,8
     ,103,230,56,192,124,214,64             [1FF0]
2640 DATA 103,125,198,80,111,208,36,124,
     230,7,192,124,214,8,103,201            [2F24]
2650 DATA 69,203,133,76,38,0,84,93,41,41
     ,25,41,41,41,89,25,237,91              [DA98]
2660 DATA 201,177,25,124,230,7,103,58,20
     3,177,132,203,24,48,2,198              [187C]
2670 DATA 32,103,201,27,60,4,14,48,117,2
     7,14,4,14,108,117,8,64,8               [3102]
2680 DATA 16,168,117,44,64,8,16,42,118,4
     0,56,8,16,172,118,40,16,8              [5196]
2690 DATA 16,46,119,16,64,8,16,176,119,1
     6,8,8,16,60,120,44,72,8,16             [08FE]
2700 DATA 190,120,28,56,8,16,64,121,28,5
     6,8,16,194,121,28,56,8,16              [BE90]
2710 DATA 68,122,15,10,8,16,198,122,0,0,
     8,16,72,123,0,0,8,16,202               [BCDA]
2720 DATA 123,27,28,4,8,152,123,24,59,4,
     8,192,123,24,32,8,16,232               [9D0E]
2730 DATA 123,20,20,8,16,232,123,19,47,1
     ,4,106,124,32,40,4,14,116              [9F34]
2740 DATA 124,0,140,32,38,48,53,44,38,48
     ,65,44,38,48,51,44,38,48               [B242]
4900 MEMORY 29999:ON ERROR GOTO 6000        [0F78]
5000 READ a                                 [1A04]
5005 READ b:IF b>255 THEN a=b:GOTO 5005     [5AF2]
5010 POKE a,b:a=a+1:GOTO 5005               [FD2C]
6000 SAVE"CHOPPER.BIN",b,30000,12000        [2F7A]
```

Listing 2. »Chopper« Basic-Lader (Schluß)

Light-Cycles



Die Idee zu diesem Spiel stammt aus dem Science-Fiction-Film »TRON«. Um gegen Ihre Freunde oder den Computer zu gewinnen, brauchen Sie blitzschnelle Reaktionen.

Denjenigen, die die Regeln von »Light-Cycles« nicht kennen, seien hier folgende Informationen gegeben: Zwei futuristische Motorräder werden automatisch beschleunigt und müssen von den Spielern innerhalb eines Spielfeldes gelenkt werden. Dabei ist unbedingt darauf zu achten, daß bereits überfahrene Stellen oder die begrenzende Wand keinesfalls berührt werden dürfen. Ziel des Spieles ist, den (oder die) Gegner durch geschicktes Manövrieren zu zwingen, die eben beschriebenen Fehler zu begehen.

Gelenkt wird mit Joysticks. Im Titelbild lassen sich verschiedene Spieloptionen über die Cursor-Steuer-Tasten wählen: Spieler 1 gegen Spieler 2, Spieler gegen Computer, Computer gegen Computer (Demo). Die Geschwindigkeit ist im Bereich von 1 bis 50 einstellbar, wobei 1 der schnellsten, 50 der langsamsten Variante entspricht. Bei den Varianten »Spieler gegen Spieler« und »Computer gegen Computer« blinkt am Spielende der Verlierer, anschließend erscheint wieder das Titelbild. Besiegt ein Spieler den Computer, ist das Spiel nicht beendet, vielmehr beginnt eine Runde mit höherem Schwierigkeitsgrad. Ab der zehnten Runde muß man gegen zwei vom Computer gesteuerte Gegner antreten (Fortgeschrittene können den Zeitpunkt dieser Steigerung festlegen, indem sie das gewünschte Level mit »POKE &7821,level« in Adresse 7821 hex eintragen).

Das gesamte Programm wurde in reiner Maschinen-Sprache geschrieben und erfüllt so hohe Qualitätsansprüche. Es nutzt die stereofone Tonerzeugung der CPCs, so daß sich der Anschluß einer Stereo-Anlage oder zumindest eines Kopfhörers empfiehlt, um den »Rennbahnsound« voll auszukosten.

Wenn Sie das Listing des Basic-Laders im Computer haben, sollten Sie es vor dem ersten Probelauf sicherheitshalber speichern. Erst danach starten Sie es wie gewohnt mit »RUN«. Da dieser Lader nur dazu dient, das Maschinenprogramm im Arbeitsspeicher zu erzeugen, beginnt nicht etwa gleich das Spiel, sondern der Computer meldet nach kurzer Zeit »Ready«. Nun können Sie das Programm mit »SAVE "Name",b,&7200,&FFF,&7200« als Binär-Datei speichern. Im Prinzip brauchen Sie jetzt den Basic-Lader nicht mehr. Löschen Sie ihn jedoch nicht, bevor Sie ganz sicher sind, daß alles korrekt funktioniert. Wenn das Maschinen-Programm mit der genannten Befehlsfolge gespeichert wurde, ist es zukünftig mit »RUN "Name"« zu starten.

(Rüdiger Görsch/ja)

```
1 DATA CD,0,BB,CD,4E,BB,CD,BA,BB,CD         [67D6]
2 DATA FF,BB,3E,1,32,E1,73,3E,14,32         [89B0]
3 DATA E2,73,3E,1,CD,E,BC,3E,0,1            [CB9A]
4 DATA 0,0,CD,32,BC,3E,0,1,0,0              [5B48]
5 DATA CD,38,BC,3E,1,1,1A,1A,CD,32          [AF78]
6 DATA BC,3E,3,CD,90,BB,21,7,9,CD           [5634]
7 DATA 75,BB,6,15,21,E8,73,C5,7E,E5         [24C2]
8 DATA CD,5A,BB,E1,23,C1,10,F5,21,14        [F624]
9 DATA 5,22,8E,80,3E,2,1,0,0,CD             [33BE]
10 DATA 32,BC,3E,3,1,2,2,CD,32,BC           [B8C4]
11 DATA 3E,2,CD,90,BB,21,19,1,CD,75         [4CAE]
12 DATA BB,6,D,21,FD,73,C5,7E,E5,CD         [B026]
13 DATA 5A,BB,E1,23,C1,10,F5,3E,0,32        [48F8]
```

Listing. Knapp 4 KByte Maschinen-Code sorgen für spannungsvolle Unterhaltung mit »Light-Cycles«

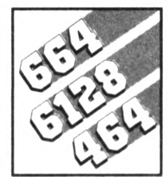
```
14 DATA E3,73,32,E5,73,CD,30,7F,21,14      [8E4A]
15 DATA 5,22,8E,80,3E,96,32,E7,73,2A       [28DE]
16 DATA E5,73,ED,5B,E3,73,CD,F0,BB,FE      [423E]
17 DATA 0,20,69,21,CE,0,ED,4B,E3,73        [F796]
18 DATA ED,42,ED,5B,E5,73,EB,CD,F0,BB      [9754]
19 DATA FE,0,C2,40,73,3A,E5,73,3C,3C       [3F24]
20 DATA FE,10,28,36,32,E5,73,3A,E7,73      [A854]
21 DATA FE,0,28,E,3D,32,E7,73,2A,8E        [0ECA]
22 DATA 80,2B,2B,22,8E,80,18,B,3E,96       [6EEA]
23 DATA 32,E7,73,21,14,5,22,8E,80,3E       [09A8]
24 DATA 0,32,8C,80,CD,12,7F,3E,2,32        [9474]
25 DATA 8C,80,CD,12,7F,C3,95,72,3E,0       [8B18]
26 DATA 32,E5,73,3A,E3,73,3C,3C,FE,68      [72A4]
27 DATA 32,E3,73,CA,9B,73,18,89,2A,E3      [5588]
28 DATA 73,A7,CB,15,CB,14,ED,4B,E3,73      [B0D0]
29 DATA 9,1,A,0,9,EB,2A,E5,73,A7          [E976]
30 DATA CB,15,CB,14,CB,15,CB,14,1,40       [4A10]
31 DATA 1,9,E5,D5,3E,1,CD,DE,BB,D1         [62B2]
32 DATA E1,CD,EA,BB,CD,76,73,C3,A3,72      [1F06]
33 DATA 2A,E3,73,A7,CB,15,CB,14,ED,4B      [79DA]
34 DATA E3,73,9,EB,21,6C,2,ED,52,EB        [FBE8]
35 DATA 2A,E5,73,A7,CB,15,CB,14,CB,15      [0CBA]
36 DATA CB,14,1,40,1,9,E5,D5,3E,3          [A2AC]
37 DATA CD,DE,BB,D1,E1,CD,EA,BB,CD,76      [8EA8]
38 DATA 73,C3,B9,72,11,3,0,21,3,0          [7C40]
39 DATA CD,F9,BB,11,FC,FF,21,3,0,CD        [B340]
40 DATA F9,BB,11,FC,FF,21,FC,FF,CD,F9      [1C84]
41 DATA BB,11,3,0,21,FC,FF,CD,F9,BB        [C42C]
42 DATA C9,21,E8,3,22,8E,80,3E,6,32        [036E]
43 DATA 8C,80,3E,1E,32,8D,80,CD,0,7F       [5C36]
44 DATA 3E,7,32,8C,80,3E,36,32,8D,80       [86DE]
45 DATA CD,0,7F,3E,0,32,8C,80,CD,12        [DABA]
46 DATA 7F,6,FA,C5,1,FF,0,0,10,FD          [7714]
47 DATA C1,10,F6,3E,0,32,8D,80,3E,6        [9382]
48 DATA 32,8C,80,CD,0,7F,CD,69,7F,18       [1654]
49 DATA 29,0,0,0,0,0,0,0,A4,20             [B3CA]
50 DATA 62,79,20,52,75,65,64,69,67,65      [6FE8]
51 DATA 72,20,47,6F,65,72,73,63,68,53      [6DF6]
52 DATA 75,70,65,72,20,2D,20,43,79,63      [40DC]
53 DATA 6C,65,21,19,1,CD,75,BB,6,E         [E444]
54 DATA C5,3E,20,CD,5A,BB,C1,10,F7,3E      [EED8]
55 DATA 2,1,6,6,CD,32,BC,3E,2,CD           [9582]
56 DATA 90,BB,21,A,8,11,5A,75,6,16         [ABF6]
57 DATA CD,45,75,6,16,21,C,8,11,70         [77E4]
58 DATA 75,CD,45,75,6,16,21,E,8,11         [5EF4]
59 DATA 86,75,CD,45,75,6,16,21,10,8        [1C46]
60 DATA 11,9C,75,CD,45,75,3E,3,CD,90       [3E1A]
61 DATA BB,21,14,1,11,B2,75,6,1F,CD        [4180]
62 DATA 45,75,6,1A,21,16,1,11,D1,75        [F112]
63 DATA CD,45,75,3E,2,CD,90,BB,6,B         [0D82]
64 DATA 21,18,A,11,EB,75,CD,45,75,3E       [BD0C]
65 DATA 1,CD,90,BB,21,18,19,CD,75,BB       [5946]
66 DATA CD,21,75,3A,E1,73,B7,17,CE,A       [844C]
67 DATA 32,59,75,6F,26,4,CD,75,BB,3E       [9932]
68 DATA E7,CD,5A,BB,CD,18,BB,FE,F0,28      [9360]
69 DATA 1C,FE,F1,28,3A,FE,F2,28,58,FE      [EB1C]
70 DATA F3,28,61,FE,20,CA,F6,75,21,18      [E278]
71 DATA 19,CD,75,BB,CD,21,75,18,C8,3A      [11BA]
72 DATA 59,75,26,4,6F,CD,75,BB,3E,20       [E224]
73 DATA CD,5A,BB,3A,E1,73,FE,0,28,6        [5500]
74 DATA 3D,32,E1,73,18,DA,3E,3,32,E1       [5402]
75 DATA 73,18,D3,3A,59,75,26,4,6F,CD       [C212]
76 DATA 75,BB,3E,20,CD,5A,BB,3A,E1,73      [18E4]
77 DATA FE,3,28,6,3C,32,E1,73,18,B8        [23A0]
78 DATA 3E,0,32,E1,73,18,B1,3A,E2,73       [E5E6]
79 DATA FE,1,28,1,3D,32,E2,73,18,A4        [0A90]
80 DATA 3A,E2,73,FE,32,28,F4,3C,18,F1      [769A]
81 DATA C9,6,0,3A,E2,73,B7,FE,A,28         [8A6A]
82 DATA 6,30,4,60,6F,18,5,D6,A,4           [6E18]
83 DATA 18,F0,7C,CE,2F,E5,CD,5A,BB,E1      [2F36]
84 DATA 7D,CE,30,CD,5A,BB,C9,C5,D5,CD      [565A]
85 DATA 75,BB,D1,C1,EB,C5,7E,E5,CD,5A      [5B3C]
86 DATA BB,E1,23,C1,10,F5,C9,0,50,4C       [3516]
87 DATA 41,59,45,52,20,31,20,20,56,53      [E6AA]
88 DATA 2E,20,50,4C,41,59,45,52,20,32      [CBEA]
89 DATA 50,4C,41,59,45,52,20,31,20,20      [2FC0]
90 DATA 56,53,2E,20,43,4F,4D,50,55,54      [4018]
91 DATA 45,52,50,4C,41,59,45,52,20,32      [55CC]
92 DATA 20,20,56,53,2E,20,43,4F,4D,50      [37FE]
93 DATA 55,54,45,52,43,4F,4D,50,55,54      [550C]
94 DATA 45,52,20,20,56,53,2E,20,43,4F      [2EE8]
95 DATA 4D,50,55,54,45,52,43,55,52,53      [36E8]
96 DATA 4F,52,20,4B,45,59,53,20,20,20      [E1E6]
97 DATA 20,20,6F,70,74,69,6F,6E,73,20      [EB34]
98 DATA 2F,20,73,70,65,65,64,53,50,41      [83E8]
99 DATA 43,45,20,42,41,52,20,20,20,20      [B988]
100 DATA 20,20,20,73,74,61,72,74,20,67     [37F8]
101 DATA 61,6D,65,53,70,65,65,64,20,6C     [1F5C]
102 DATA 65,76,65,6C,3A,E1,73,FE,0,28      [C35C]
103 DATA 10,FE,1,CA,2F,77,FE,2,CA,3B       [386E]
104 DATA 77,FE,3,CA,B8,76,C9,CD,B7,78      [C3FA]
105 DATA CD,30,7F,CD,A0,80,3E,3,32,12      [316C]
106 DATA 90,3E,4,32,1A,90,3E,1E,32,10      [FE0C]
107 DATA 90,3E,7D,32,18,90,3E,50,32,11     [CD6A]
108 DATA 90,32,19,90,CD,2C,80,3E,0,32      [0422]
109 DATA 90,80,CD,EF,78,3E,2,CD,DE,BB      [7320]
110 DATA CD,0,7D,CD,E7,7F,CD,44,80,CD      [C110]
111 DATA 38,80,3E,2,32,90,80,CD,EF,78      [9C5A]
112 DATA 3E,3,CD,DE,BB,CD,0,7D,CD,E7       [F1DC]
113 DATA 7F,CD,50,80,3A,14,90,FE,1,28      [1568]
114 DATA 16,3A,1C,90,FE,1,28,1F,3A,E2      [3F78]
115 DATA 73,47,C5,6,FF,0,10,FD,C1,10       [5FF8]
116 DATA F7,18,B1,3E,2,1,5,0,CD,32         [5206]
117 DATA BC,CD,3E,78,CD,69,7F,18,E,3E      [DA16]
118 DATA 3,1,5,0,CD,32,BC,CD,3E,78         [FE50]
119 DATA CD,69,7F,6,5,FB,C5,6,FF,C5        [8B32]
120 DATA 6,FF,0,10,FD,C1,10,F7,C1,10       [B8FA]
121 DATA F1,F3,CD,0,BB,C3,16,72,CD,B7      [88D6]
122 DATA 78,CD,30,7F,CD,A0,80,21,1,4       [51F2]
123 DATA CD,AC,78,32,22,90,CD,AC,78,32     [FD0E]
124 DATA 2A,90,21,A,50,CD,AC,78,32,20      [E946]
125 DATA 90,21,5A,96,CD,AC,78,32,28,90     [E0C2]
126 DATA 21,14,8C,CD,AC,78,32,21,90,CD     [B7E8]
127 DATA AC,78,32,29,90,CD,5C,80,3E,0      [A876]
128 DATA 32,91,80,3E,2,CD,DE,BB,CD,0       [D05A]
129 DATA 7A,CD,A2,7F,CD,74,80,CD,68,80     [3D50]
130 DATA 3E,2,32,91,80,3E,3,CD,DE,BB       [3A34]
131 DATA CD,0,7A,CD,A2,7F,CD,80,80,3A      [DAD8]
132 DATA 24,90,FE,1,CA,81,76,3A,2C,90      [6966]
133 DATA FE,1,CA,91,76,18,C2,3E,0,CD       [3042]
134 DATA EF,78,3E,0,32,90,80,18,A,3E       [83F0]
135 DATA 2,CD,EF,78,3E,0,32,90,80,3E       [2C10]
136 DATA 1,32,32,90,3E,0,32,30,90,32       [F044]
137 DATA 31,90,32,4,90,3E,7D,32,0,90       [B982]
138 DATA 3E,50,32,1,90,3E,1,32,2,90        [1B0E]
139 DATA 3E,19,32,20,90,3E,64,32,21,90     [4350]
140 DATA 3E,3,32,22,90,3E,4B,32,28,90      [25FC]
141 DATA 3E,1E,32,29,90,3E,1,32,2A,90      [441A]
142 DATA CD,B7,78,CD,30,7F,CD,A0,80,3E     [7F56]
143 DATA 2,CD,DE,BB,CD,0,7D,CD,E7,7F       [D8EC]
144 DATA 3A,4,90,FE,0,28,3,C3,81,76        [DF68]
145 DATA 3E,3,CD,DE,BB,3A,32,90,FE,1       [407A]
146 DATA 28,25,3A,31,90,FE,1,28,1E,CD      [686C]
147 DATA 68,80,CD,0,7A,3E,2,32,91,80       [95D0]
148 DATA CD,A2,7F,CD,80,80,3A,2C,90,32     [0F10]
149 DATA 31,90,FE,0,28,3,CD,3E,78,3A       [4316]
150 DATA 30,90,FE,1,28,1E,CD,5C,80,CD      [67A4]
151 DATA 0,7A,3E,2,32,91,80,CD,A2,7F       [1EFA]
152 DATA CD,74,80,3A,24,90,32,30,90,FE     [3CAE]
153 DATA 1,20,3,CD,3E,78,3A,32,90,FE       [1000]
154 DATA 2,28,9,3A,30,90,FE,1,28,19        [0E4A]
155 DATA 18,7,3A,31,90,FE,1,28,F0,3A       [0EE4]
156 DATA E2,73,47,C5,6,FF,0,10,FD,C1       [762E]
157 DATA 10,F7,C3,8B,77,3A,E2,73,FE,1      [68A0]
158 DATA 28,7,3D,32,E2,73,C3,4A,77,3A      [B25E]
159 DATA 32,90,FE,2,28,4,3C,32,32,90       [28AE]
160 DATA 3E,A,32,E2,73,C3,4A,77,3E,6       [2006]
161 DATA 32,8C,80,3E,F,32,8D,80,CD,0       [C808]
162 DATA 7F,3E,0,32,8D,80,3E,7,32,8C       [76F8]
163 DATA 80,CD,0,7F,3E,F,32,8D,80,3E       [9D36]
164 DATA 9,32,8C,80,CD,0,7F,21,E8,3        [2092]
165 DATA 22,8E,80,3E,4,32,8C,80,CD,12      [3152]
166 DATA 7F,6,4,C5,6,FF,C5,6,FF,0          [DA1C]
167 DATA 10,FD,C1,10,F7,C1,10,F1,3E,38     [EFE0]
168 DATA 32,8D,80,3E,7,32,8C,80,CD,0       [02F8]
169 DATA 7F,3E,0,32,8D,80,3E,4,32,8C       [6C00]
170 DATA 80,CD,0,7F,3E,0,32,8D,80,3E       [6C06]
171 DATA 9,32,8C,80,CD,0,7F,C9,ED,5F       [BF68]
172 DATA 1F,1F,BD,38,F9,BC,30,F6,C9,3E     [4A6C]
173 DATA 3C,32,15,90,32,1D,90,32,25,90     [EB46]
174 DATA 32,2D,90,21,14,5,22,16,90,22      [C1B6]
175 DATA 1E,90,22,26,90,22,2E,90,3E,0      [4206]
176 DATA 32,13,90,32,1B,90,32,23,90,32     [421E]
177 DATA 2B,90,32,14,90,32,1C,90,32,24     [CA44]
178 DATA 90,32,2C,90,C9,FE,0,28,6,3E       [B402]
179 DATA 7D,32,3,7D,C9,3E,7C,32,3,7D       [162C]
180 DATA C9,0,CD,17,79,CD,29,79,CD,F0      [D4D2]
181 DATA BB,FE,5,28,7,FE,0,28,3,3E         [476A]
182 DATA 1,C9,3E,0,C9,A7,2A,A5,79,7E       [053E]
183 DATA 11,0,0,5F,CB,13,CB,12,CB,13       [1CF2]
184 DATA CB,12,C9,A7,2A,A7,79,7E,21,0      [7E92]
185 DATA 0,6F,CB,15,CB,14,C9,21,9,90       [A10C]
186 DATA 34,C9,21,8,90,34,C9,21,8,90       [8EA6]
187 DATA 35,C9,21,9,90,35,C9,3A,AD,79      [A878]
188 DATA 21,AE,79,B6,C9,3A,AA,79,FE,0      [7CEA]
189 DATA 28,1B,3A,A9,79,FE,1,28,3,3E       [6428]
190 DATA 1,C9,3A,AD,79,FE,1,28,3,3E        [A8CE]
191 DATA 0,C9,3E,1,32,C,90,18,F6,3A        [6380]
192 DATA A9,79,FE,0,28,3,3E,2,C9,3A        [F7BC]
193 DATA AC,79,FE,0,28,15,3A,AB,79,FE      [C7E8]
194 DATA 1,28,3,3E,1,C9,ED,5F,1F,38        [DBBA]
195 DATA 3,3E,1,C9,3E,2,C9,3A,AB,79        [60D4]
196 DATA FE,0,28,EE,3E,2,C9,0,0,0          [FED6]
197 DATA 0,0,0,0,0,0,0,21,8,90             [4DB6]
198 DATA 22,A5,79,21,9,90,22,A7,79,CD      [F954]
199 DATA 17,79,CD,29,79,CD,EA,BB,C9,0      [1900]
200 DATA 0,0,0,0,0,0,0,0,0,0               [B5B0]
201 DATA 0,0,0,0,0,0,0,0,0,0               [DEB2]
202 DATA 0,0,0,0,0,0,0,0,0,0               [4BB4]
203 DATA 0,0,0,0,0,0,0,0,0,0               [E4B6]
204 DATA 0,0,0,0,0,0,0,0,0,0               [21B8]
205 DATA 0,0,0,0,0,0,0,0,3A,A              [2D64]
206 DATA 90,FE,1,28,D,FE,2,CA,89,7A        [7BF6]
207 DATA FE,3,CA,FE,7A,C3,73,7B,CD,37      [C71C]
208 DATA 79,CD,E8,7B,CD,0,79,32,AD,79      [90DC]
209 DATA CD,2,7C,CD,0,79,32,AE,79,CD       [0C7A]
210 DATA 4B,79,FE,0,CA,AF,79,CD,E8,7B      [D024]
211 DATA CD,C,7C,CD,FD,7B,CD,0,79,32       [C4A8]
212 DATA A9,79,CD,FD,7B,CD,0,79,32,AB      [B5FC]
213 DATA 79,CD,E8,7B,CD,7,7C,CD,C,7C       [7CCA]
214 DATA CD,0,79,32,AA,79,CD,7,7C,CD       [7074]
```

Listing. »Light-Cycles« (Fortsetzung)

```
215 DATA 0,79,32,AC,79,CD,53,79,FE,0          [6322]
216 DATA CA,AF,79,FE,2,28,E,CD,3C,79          [9FA6]
217 DATA CD,46,79,3E,3,32,A,90,C3,AF          [AF30]
218 DATA 79,CD,41,79,CD,46,79,3E,4,32         [177E]
219 DATA A,90,C3,AF,79,CD,46,79,CD,E8         [93FC]
220 DATA 7B,CD,0,79,32,AD,79,CD,C,7C          [9670]
221 DATA CD,0,79,32,AE,79,CD,4B,79,FE         [C6EA]
222 DATA 0,CA,AF,79,CD,E8,7B,CD,2,7C          [00AE]
223 DATA CD,7,7C,CD,0,79,32,A9,79,CD          [2364]
224 DATA 7,7C,CD,0,79,32,AB,79,CD,E8          [2864]
225 DATA 7B,CD,2,7C,CD,FD,7B,CD,0,79          [54B8]
226 DATA 32,AA,79,CD,FD,7B,CD,0,79,32         [DEDA]
227 DATA AC,79,CD,53,79,FE,0,CA,AF,79         [2D14]
228 DATA FE,2,28,E,CD,37,79,CD,41,79          [8A56]
229 DATA 3E,4,32,A,90,C3,AF,79,CD,37          [F038]
230 DATA 79,CD,3C,79,3E,3,32,A,90,C3          [8810]
231 DATA AF,79,CD,3C,79,CD,E8,7B,CD,0         [4928]
232 DATA 79,32,AD,79,CD,FD,7B,CD,0,79         [31F0]
233 DATA 32,AE,79,CD,4B,79,FE,0,CA,AF         [FA18]
234 DATA 79,CD,E8,7B,CD,7,7C,CD,C,7C          [1DD0]
235 DATA CD,0,79,32,A9,79,CD,C,7C,CD          [8382]
236 DATA 0,79,32,AB,79,CD,E8,7B,CD,7          [F768]
237 DATA 7C,CD,2,7C,CD,0,79,32,AA,79          [1F5A]
238 DATA CD,2,7C,CD,0,79,32,AC,79,CD          [B77A]
239 DATA 53,79,FE,0,CA,AF,79,FE,2,28          [5F72]
240 DATA E,CD,41,79,CD,46,79,3E,2,32          [DD1A]
241 DATA A,90,C3,AF,79,CD,41,79,CD,37         [69C2]
242 DATA 79,3E,1,32,A,90,C3,AF,79,CD          [6C34]
243 DATA 41,79,CD,E8,7B,CD,0,79,32,AD         [EDC4]
244 DATA 79,CD,7,7C,CD,0,79,32,AE,79          [0254]
245 DATA CD,4B,79,FE,0,CA,AF,79,CD,E8         [4D50]
246 DATA 7B,CD,FD,7B,CD,2,7C,CD,0,79          [72BE]
247 DATA 32,A9,79,CD,2,7C,CD,0,79,32          [EA22]
248 DATA AB,79,CD,E8,7B,CD,FD,7B,CD,C         [8290]
249 DATA 7C,CD,0,79,32,AA,79,CD,C,7C          [2D82]
250 DATA CD,0,79,32,AC,79,CD,53,79,FE         [E4CE]
251 DATA 0,CA,AF,79,FE,2,28,E,CD,3C           [3224]
252 DATA 79,CD,37,79,3E,1,32,A,90,C3          [1FFC]
253 DATA AF,79,CD,3C,79,CD,46,79,3E,2         [4ADE]
254 DATA 32,A,90,C3,AF,79,2A,8,90,22          [16EA]
255 DATA FB,7B,21,FB,7B,22,A5,79,21,FC        [C03A]
256 DATA 7B,22,A7,79,C9,0,0,21,FB,7B          [2610]
257 DATA 34,C9,21,FC,7B,34,C9,21,FB,7B        [CA26]
258 DATA 35,C9,21,FC,7B,35,C9,0,0,0           [D984]
259 DATA 0,0,0,0,0,0,0,0,0,0                  [BCCC]
260 DATA 0,0,0,0,0,0,0,0,0,0                  [55BC]
261 DATA 0,0,0,0,0,0,0,0,0,0                  [BEBE]
262 DATA 0,0,0,0,0,0,0,0,0,0                  [1FC0]
263 DATA 0,0,0,0,0,0,0,0,0,0                  [D8C2]
264 DATA 0,0,0,0,0,0,0,0,0,0                  [49C4]
265 DATA 0,0,0,0,0,0,0,0,0,0                  [E2C6]
266 DATA 0,0,0,0,0,0,0,0,0,0                  [B3C8]
267 DATA 0,0,0,0,0,0,0,0,0,0                  [DCCA]
268 DATA 0,0,0,0,0,0,0,0,0,0                  [BDCC]
269 DATA 0,0,0,0,0,0,0,0,0,0                  [06CE]
270 DATA 0,0,0,0,0,0,0,0,0,0                  [DDBE]
271 DATA 0,0,0,0,0,0,0,0,0,0                  [B4C0]
272 DATA 0,0,0,0,0,0,0,0,0,0                  [E3C2]
273 DATA 0,0,0,0,0,0,0,0,0,0                  [4AC4]
274 DATA 0,0,0,0,0,0,0,0,0,0                  [D9C6]
275 DATA 0,0,0,0,0,0,0,0,0,0                  [20C8]
276 DATA 0,0,0,0,0,0,0,0,0,0                  [BFCA]
277 DATA 0,0,0,0,0,0,0,0,0,0                  [56CC]
278 DATA 0,0,0,0,0,0,0,0,0,0                  [E5CE]
279 DATA 0,0,0,0,0,0,0,0,0,0                  [DCD0]
280 DATA 0,0,0,0,0,0,0,0,0,0                  [E5C0]
281 DATA 0,0,0,0,0,0,0,0,0,0                  [EEC2]
282 DATA 0,0,0,0,0,0,CD,24,BB,7C              [4F1A]
283 DATA CB,A7,FE,0,20,3,3A,2,90,32           [F188]
284 DATA 2,90,CB,47,20,E,CB,4F,20,10          [40F0]
285 DATA CB,57,20,12,CB,5F,20,14,18,E8        [E3D0]
286 DATA 21,1,90,34,18,10,21,1,90,35          [EC32]
287 DATA 18,A,21,0,90,35,18,4,21,0            [D08A]
288 DATA 90,34,A7,21,0,90,7E,11,0,0           [B014]
289 DATA 5F,CB,13,CB,12,CB,13,CB,12,A7        [623C]
290 DATA 21,1,90,7E,21,0,0,6F,CB,15           [F448]
291 DATA CB,14,D5,E5,CD,F0,BB,FE,0,28         [420A]
292 DATA F,FE,5,28,B,E1,D1,CD,EA,BB           [0662]
293 DATA 3E,1,32,4,90,C9,E1,D1,CD,EA          [4B46]
294 DATA BB,C9,0,0,0,0,0,0,0,0                [5B0A]
295 DATA 0,0,0,0,0,0,0,0,0,0                  [C8CC]
296 DATA 0,0,0,0,0,0,0,0,0,0                  [2BCE]
297 DATA 0,0,0,0,0,0,0,0,0,0                  [32D0]
298 DATA 0,0,0,0,0,0,0,0,0,0                  [EDD2]
299 DATA 0,0,0,0,0,0,0,0,0,0                  [54D4]
300 DATA 0,0,0,0,0,0,0,0,0,0                  [DCB2]
301 DATA 0,0,0,0,0,0,0,0,0,0                  [B3B4]
302 DATA 0,0,0,0,0,0,0,0,0,0                  [E2B6]
303 DATA 0,0,0,0,0,0,0,0,0,0                  [49B8]
304 DATA 0,0,0,0,0,0,0,0,0,0                  [D8BA]
305 DATA 0,0,0,0,0,0,0,0,0,0                  [1FBC]
306 DATA 0,0,0,0,0,0,0,0,0,0                  [BEBE]
307 DATA 0,0,0,0,0,0,0,0,0,0                  [55C0]
308 DATA 0,0,0,0,0,0,0,0,0,0                  [E4C2]
309 DATA 0,0,0,0,0,0,0,0,0,0                  [DBC4]
310 DATA 0,0,0,0,0,0,0,0,0,0                  [54B4]
311 DATA 0,0,0,0,0,0,0,0,0,0                  [BDB6]
312 DATA 0,0,0,0,0,0,0,0,0,0                  [1EB8]
313 DATA 0,0,0,0,0,0,0,0,0,0                  [D7BA]
314 DATA 0,0,0,0,0,0,0,0,0,0                  [48BC]
315 DATA 0,0,0,0,0,0,0,0,0,0                  [E1BE]
316 DATA 0,0,0,0,0,0,0,0,0,0                  [B2C0]
317 DATA 0,0,0,0,0,0,0,0,0,0                  [DBC2]
318 DATA 0,0,0,0,0,0,0,0,0,0                  [BCC4]
319 DATA 0,0,0,0,0,0,0,0,0,0                  [05C6]
320 DATA 0,0,0,0,0,0,0,0,0,0                  [BCB6]
321 DATA 0,0,0,0,0,0,0,0,0,0                  [53B8]
322 DATA 0,0,0,0,0,0,0,0,0,0                  [D6BA]
323 DATA 0,0,0,0,0,0,0,0,0,0                  [1DBC]
324 DATA 0,0,0,0,0,0,0,0,0,0                  [E0BE]
325 DATA 0,0,0,0,0,0,0,0,0,0                  [47C0]
326 DATA 0,0,0,0,0,0,0,0,0,0                  [DAC2]
327 DATA 0,0,0,0,0,0,0,0,0,0                  [B1C4]
328 DATA 0,0,0,0,0,0,0,0,0,0                  [04C6]
329 DATA 0,0,0,0,0,0,0,0,0,0                  [BBC8]
330 DATA 0,0,0,0,0,0,0,0,0,0                  [1CB8]
331 DATA 0,0,0,0,0,0,0,0,0,0                  [D5BA]
332 DATA 0,0,0,0,0,0,0,0,0,0                  [52BC]
333 DATA 0,0,0,0,0,0,0,0,CD,6                 [3A78]
334 DATA B9,3A,8C,80,21,8D,80,4E,CD,26        [0102]
335 DATA 8,CD,9,B9,F3,C9,CD,6,B9,3A           [862A]
336 DATA 8C,80,21,8E,80,4E,CD,26,8,3A         [4B82]
337 DATA 8C,80,3C,21,8E,80,23,4E,CD,26        [41E2]
338 DATA 8,CD,9,B9,F3,C9,3E,7,32,8C           [DDF6]
339 DATA 80,3E,38,32,8D,80,CD,0,7F,3E         [898A]
340 DATA 8,32,8C,80,3E,F,32,8D,80,CD          [EF16]
341 DATA 0,7F,3E,9,32,8C,80,CD,0,7F           [6EB0]
342 DATA 21,0,0,22,8E,80,3E,0,32,8C           [D726]
343 DATA 80,CD,12,7F,3E,2,32,8C,80,CD         [E790]
344 DATA 12,7F,C9,3E,7,32,8C,80,3E,3F         [828A]
345 DATA 32,8D,80,CD,0,7F,3E,0,32,8C          [980E]
346 DATA 80,21,0,0,22,8E,80,CD,12,7F          [38BC]
347 DATA 3E,2,32,8C,80,CD,12,7F,3E,0          [F50A]
348 DATA 32,8D,80,3E,8,32,8C,80,CD,0          [EAFA]
349 DATA 7F,3E,9,32,8C,80,CD,0,7F,C9          [3D58]
350 DATA 3A,A,90,21,B,90,BE,20,29,3A          [1CE4]
351 DATA D,90,FE,0,28,12,3D,32,D,90           [E97A]
352 DATA 2A,E,90,2B,2B,2B,2B,2B,22,E          [241C]
353 DATA 90,22,8E,80,2A,E,90,22,8E,80         [953A]
354 DATA 3A,91,80,32,8C,80,CD,12,7F,C9        [A8E6]
355 DATA 3E,3C,32,D,90,21,14,5,22,E           [4152]
356 DATA 90,3A,A,90,32,B,90,18,DD,3A          [0D00]
357 DATA 2,90,21,3,90,BE,20,29,3A,5           [4E38]
358 DATA 90,FE,0,28,12,3D,32,5,90,2A          [64C8]
359 DATA 6,90,2B,2B,2B,2B,2B,22,6,90          [B6DA]
360 DATA 22,8E,80,2A,6,90,22,8E,80,3A         [D42E]
361 DATA 90,80,32,8C,80,CD,12,7F,C9,3E        [3BE8]
362 DATA 3C,32,5,90,21,14,5,22,6,90           [19F4]
363 DATA 3A,2,90,32,3,90,18,DD,21,10          [AB8E]
364 DATA 90,11,0,90,1,8,0,ED,B0,C9            [3CEE]
365 DATA 21,18,90,11,0,90,1,8,0,ED            [7EAC]
366 DATA B0,C9,21,0,90,11,10,90,1,8           [0B08]
367 DATA 0,ED,B0,C9,21,0,90,11,18,90          [E4BA]
368 DATA 1,8,0,ED,B0,C9,21,20,90,11           [734E]
369 DATA 8,90,1,8,0,ED,B0,C9,21,28            [560C]
370 DATA 90,11,8,90,1,8,0,ED,B0,C9            [42F8]
371 DATA 21,8,90,11,20,90,1,8,0,ED            [C6A8]
372 DATA B0,C9,21,8,90,11,28,90,1,8           [A224]
373 DATA 0,ED,B0,C9,0,0,0,0,0,0               [9D94]
374 DATA 0,0,0,0,0,0,0,0,0,0                  [20C8]
375 DATA 0,0,0,0,CD,0,BB,CD,4E,BB             [2508]
376 DATA CD,BA,BB,CD,FF,BB,3E,0,CD,E          [753E]
377 DATA BC,3E,2,CD,E4,BB,3E,1,CD,38          [8BAA]
378 DATA BC,3E,4,1,6,6,CD,32,BC,3E            [607E]
379 DATA 5,1,D,D,CD,32,BC,3E,0,1              [9F82]
380 DATA 0,0,CD,32,BC,3E,1,1,1A,1A            [3820]
381 DATA CD,32,BC,3E,1,CD,DE,BB,21,0          [7B86]
382 DATA 0,11,0,0,CD,EA,BB,11,7F,2            [D42E]
383 DATA 21,0,0,CD,F6,BB,21,8F,1,11           [2E84]
384 DATA 7F,2,CD,F6,BB,11,0,0,21,8F           [5DBC]
385 DATA 1,CD,F6,BB,11,0,0,21,0,0             [0D86]
386 DATA CD,F6,BB,21,14,0,11,13,0,CD          [5E06]
387 DATA EA,BB,21,14,0,11,6B,2,CD,F6          [1632]
388 DATA BB,21,54,1,11,6B,2,CD,F6,BB          [EA3A]
389 DATA 21,54,1,11,13,0,CD,F6,BB,21          [C9CE]
390 DATA 14,0,11,13,0,CD,F6,BB,21,0           [B04E]
391 DATA 0,11,0,0,CD,F6,BB,21,14,0            [5FE8]
392 DATA 11,6B,2,CD,EA,BB,21,0,0,11           [B68C]
393 DATA 7F,2,CD,F6,BB,21,54,1,11,6B          [7A24]
394 DATA 2,CD,EA,BB,21,68,1,11,7F,2           [4AB8]
395 DATA CD,F6,BB,21,54,1,11,13,0,CD          [D610]
396 DATA EA,BB,21,68,1,11,0,0,CD,F6           [C2B2]
397 DATA BB,11,7F,2,21,68,1,CD,F6,BB          [B34E]
398 DATA 3E,5,CD,DE,BB,1,14,0,21,27           [01BA]
399 DATA 0,22,EE,81,3E,0,32,F0,81,2A          [05E2]
400 DATA EE,81,C5,E5,EB,D5,21,16,0,CD         [F5C0]
401 DATA EA,BB,D1,21,52,1,CD,F6,BB,EB         [8106]
402 DATA E1,C1,9,22,EE,81,3A,F0,81,3C         [FF7E]
403 DATA 32,F0,81,FE,1D,20,DA,21,28,0         [B24A]
404 DATA 22,EE,81,3E,0,32,F0,81,1,14          [BBAE]
405 DATA 0,2A,EE,81,C5,E5,11,16,0,CD          [C40E]
406 DATA EA,BB,E1,E5,11,67,2,CD,F6,BB         [0F04]
407 DATA E1,C1,9,22,EE,81,3A,F0,81,3C         [6588]
408 DATA 32,F0,81,FE,F,20,DC,C9,0,0           [7CB8]
409 DATA 0,0,0,0,0,0,0,0,0,0                  [90C6]
410 DATA 0,0,0,0,0,0,0,0,0,0                  [27B6]
411 FOR i= 29184 TO 33279:READ a$:POKE i
    ,VAL("&"+a$):NEXT i                       [C8BE]
```

Listing. »Light-Cycles« (Schluß)

Zwei Strich backbord



Was dem einen sein Flugsimulator, ist dem anderen der »Schnellbootkommandant«. Verdienen Sie sich Ihre Loorbeeren auf den sieben Meeren.

Ziel des Spiels ist es, in der Rangliste vom Oberleutnant bis zum Admiral aufzusteigen. Dazu muß eine bestimmte Anzahl gegnerischer Schiffe »dran glauben«. Sollten Sie selber aber Opfer eines feindlichen Torpedos werden oder vorher aufgeben, erscheint Ihr bis dahin erreichter Dienstgrad mit Abzeichen auf dem Monitor.

Die Bewaffnung besteht aus zwei Torpedorohren und zwei Geschütztürmen. An Munition führen Sie zehn Torpedos und etwa 470 Schuß Granaten mit. Natürlich richten Torpedotreffer mehr Schaden an als Granaten. Ihr Schnellboot ist bis zu einer Beschädigung von etwa 70 Prozent fahrtüchtig.

Trifft der Gegner während eines Angriffs, leuchtet der Bildschirmrand auf. Liegt der Treffer besonders ungünstig, werden eventuell Geschütze oder Torpedorohre beschädigt. Fallen beispielsweise beide Geschütze aus, sind natürlich nur noch Torpedos einsatzfähig.

Ist ein kompletter Verband zerstört, werden alle Beschädigungen repariert (auch Geschütze und Torpedorohre) und Treibstoff und Munition aufgefüllt.

Um zwischen den Angriffen neue Vorräte zu übernehmen, kann man einen Versorger ansteuern. Anstatt des Radarbildes ist dann das Meer und der Versorger zu sehen. Alles andere geschieht automatisch. Reparaturen nimmt das Versorgungsschiff allerdings nicht vor.

Der Joystick steuert den Kurs des Bootes. Mit dem Feuerknopf wird ein Schuß aus einem der Geschütze abgegeben. Lage und Höhe der Explosion sind zufällig.

Die SPACE-Taste löst ein Torpedo aus. Da man einen Unterwasserschuß nicht sieht, bestätigt dies die Meldung »Torpedo läuft«. Ein neues Torpedo ist erst dann einsatzfähig, wenn das alte getroffen oder sein Ziel verfehlt hat (das heißt, die Meldung »Torpedo läuft« gelöscht ist).

Mit den Cursor-Tasten kann die Geschwindigkeit verändert, mit CTRL ein Angriff abgebrochen werden; es erscheint daraufhin wieder das Radarbild. CTRL legt also eine Pause im Kampfgeschehen ein, oder aber zeigt die Aufgabe des Kampfes an, sprich das Ende des Spiels.

(Rainer Krotz/ja)

```
10 REM TITELBILD                              [9144]
20 REM                                        [FBCC]
30 MODE 1:BORDER 3:INK 0,11:INK 1,1:INK
   2,0:INK 3,15:PAPER 0                       [9EAA]
40 WINDOW #1,1,40,21,25:PAPER #1,1:CLS:C
   LS #1                                      [931C]
50 MOVE 0,0:DRAW 0,399,3:DRAW 639,399:DR
   AW 639,0:DRAW 0,0                          [C9CA]
60 MOVE 528,100:DRAW 522,78,2                 [FBA6]
70 READ a,b:IF a=1 THEN 90                    [5E84]
80 DRAW a,b,2:GOTO 70                         [473A]
90 DEG:FOR a=1 TO 360 STEP 7                  [D830]
100 PLOT 365+10*COS(a),164+10*SIN(a):NEX
    T                                         [711A]
110 PEN 1:POKE &B28F,63                       [0438]
120 s$="SCHNELLBOOTKOMMANDANT"                [92E4]
130 FOR m=1 TO 21:LOCATE 10+m,5:PRINT MI
    D$(s$,m,1):SOUND 1,0,2,5,0,0,5:FOR w
    t=1 TO 90:NEXT:NEXT                       [3656]
140 POKE &B28F,255                            [1AC8]
150 PEN 2:LOCATE 11,8:PRINT "Autor: *RAI
    NER KROTZ*":PEN 1:LOCATE 17,10:PRINT
     CHR$(164);" in 1985"                     [94B0]
160 MOVE 152,312:DRAW 152,345,3:DRAW 500
    ,345:DRAW 500,312:DRAW 152,312            [ECAA]
170 FOR m=1 TO 27:READ s:SOUND 1,80,s,5:
    SOUND 1,0,10,0:NEXT                       [6492]
180 PAPER 1:PEN 3:LOCATE 9,24:PRINT "Das
     Programm wird geladen !":RUN "!Teil
    2"                                        [EFE4]
190 DATA 250,78,210,110,315,100,530,100,
    494,104,490,110,506,114,505,116,490,
    112,490,114,470,114,470,113,467,113,
    467,108,470,108,468,100,450,100,450,
    112,430,112,430,110,422,110,416,115,
    423,126,416,115,410,115,410,120,392,
    120,390,117,388,120,380,120               [8806]
200 DATA 370,150,374,150,374,153,356,153
    ,356,150,360,150,350,120,348,130,346
    ,130,346,134,340,134,340,130,310,130
    ,300,103,290,104,288,112,291,112,291
    ,116,287,116,286,118,270,118,254,122
    ,252,120,270,116,266,105,1,1              [67CA]
210 DATA 9,18,9,9,18,9,9,18,9,9,9,18,9,1
    8,9,18,9,18,9,18,18,18,18,18,18,9,9       [28B0]
220 END                                       [9B16]
```

Listing 1. Titelbild

```
10 REM *** Autor: RAINER KROTZ ***            [9742]
20 REM                                        [FBCC]
30 MODE 2:PAPER 0:PEN 1:INK 0,0:INK 1,24
                                              [EC62]
40 PRINT CHR$(22)CHR$(0)                      [76D2]
50 ENT 1,50,-10,2                             [B3CC]
60 DIM f$(3):DIM des$(6):DIM u$(8):DIM k
   (2):DIM ens(4)                             [CA40]
70 xpo=30:ypo=2:cou=2:spe=1:bil=1:fue=46
   3:fuz=0:fra=1:frt=0:frz=1200:tur=2:bu
   l=174:tor=11:tub=2:z=5                     [0D20]
80 f$(1)="{2 SPACE}ijk l "                    [877E]
90 f$(2)="u mnopqr"                           [0B92]
100 f$(3)="vssssst"                           [A5B8]
110 kla$="wxyz{"                              [C9D8]
120 rank$="OBERLEUTNANT"                      [9D2E]
130 lev=0:schi=0                              [956C]
140 MODE 1:BORDER 2                           [3AEA]
150 GOSUB 2670                                [F64A]
160 GOSUB 2180                                [2D44]
170 INK 0,0:INK 1,1:INK 2,13:INK 3,9          [C0C4]
180 WINDOW #1,14,39,2,15:PAPER #1,1           [5542]
190 WINDOW #2,2,12,2,7:PAPER #2,0             [2170]
200 WINDOW #3,32,39,22,24:PAPER #3,2          [B6A2]
210 WINDOW #4,14,39,2,11:PAPER #4,2           [A03C]
220 PAPER 3:CLS:CLS #1:CLS #2:CLS #3          [429C]
230 PAPER 2:LOCATE 14,18:PRINT STRING$(2
    6," ")                                    [3D6C]
240 PAPER 0:PEN 2:LOCATE 2,2:PRINT "ARMA
    MENT"                                     [0A1E]
250 LOCATE 2,4:PRINT "TURRET 1 +":LOCATE
     2,5:PRINT "TURRET 2 +":LOCATE 2,6:P
    RINT "TUBE{3 SPACE}1 +":LOCATE 2,7:P
    RINT "TUBE{3 SPACE}2 +"                   [46BA]
260 PAPER 3:LOCATE 2,21:PRINT "COURSE":L
    OCATE 2,24:PRINT "DAMAGE 00 %":LOCAT
    E 14,24:PRINT "FUEL"                      [C92E]
270 LOCATE 19,22:PRINT "SPEED 1{2 SPACE}
    KN"                                       [60A2]
280 LOCATE 14,22:PRINT STRING$(3,CHR$(23
    1))                                       [7202]
290 LOCATE 2,22:PRINT "SOUTH"                 [6926]
300 PEN 0:LOCATE 2,10:PRINT "BULLETS":LO
    CATE 2,13:PRINT "TORPEDOS":LOCATE 2,
    16:PRINT "RANGE -- M":LOCATE 2,18:PR
    INT "TARGET"                              [D71A]
310 PAPER 2:PEN 3:LOCATE 32,22:PRINT "PO
    SITION"                                   [4F2A]
320 LOCATE 32,23:PRINT "f ";xpo:LOCATE 3
    2,24:PRINT CHR$(94);" ";ypo               [404E]
330 LOCATE 2,11:PRINT STRING$(10," "):LO
    CATE 2,14:PRINT STRING$(10," ")           [0DD6]
340 PAPER 0:LOCATE 19,24:PRINT "{2 SPACE
    }";:PAPER 2:PRINT STRING$(9," ")          [E666]
350 MOVE 0,0:DRAW 0,399,2:DRAW 639,399:D
    RAW 639,0:DRAW 0,0                        [0B2E]
360 PLOT 15,286:DRAW 15,384:DRAW 192,384
    :DRAW 192,286:DRAW 15,286                 [7C38]
370 PLOT 206,384:DRAW 624,384:DRAW 624,1
    58:DRAW 206,158:DRAW 206,384              [105C]
380 PLOT 15,262:DRAW 15,38:DRAW 192,38:D
    RAW 192,262:DRAW 15,262                   [5548]
390 PLOT 15,38:DRAW 15,8:DRAW 192,8:DRAW
     192,38                                   [125A]
400 MOVE 15,86:DRAW 192,86:MOVE 15,136:D
    RAW 192,136:MOVE 15,166:DRAW 192,166
    :MOVE 15,216:DRAW 192,216                 [0684]
410 MOVE 202,8:DRAW 472,8:DRAW 472,38:DR
    AW 202,38:DRAW 202,8                      [11F8]
420 MOVE 202,38:DRAW 202,70:DRAW 472,70:
    DRAW 472,38                               [61B6]
430 MOVE 274,38:DRAW 274,70                   [2FF4]
440 MOVE 206,128:DRAW 624,128,0:DRAW 624
    ,110:DRAW 206,110:DRAW 206,128            [1CB8]
450 MOVE 494,14:DRAW 624,14:DRAW 624,64:
    DRAW 494,64:DRAW 494,14                   [6654]
```

Listing 2. Eher eine Simulation als ein Spiel

```
460 bil=1:CLS #1:LOCATE 8,16:PAPER 3:PEN
     0:PRINT "-- ":LOCATE 2,19:PRINT "--
    --------":LOCATE 14,18:PAPER 2:PRINT
     STRING$(25," "):MOVE 220,384:RESTOR
    E 3850                                  [B444]
470 READ a,b                                [EFCC]
480 IF a=1 THEN GOTO 510                    [2E30]
490 DRAW a,b,2                              [77B0]
500 GOTO 470                                [7752]
510 PAPER 1:PEN 2:LOCATE xpo,ypo:PRINT "
    d"                                      [33D4]
520 PEN 3                                   [2BDA]
530 IF f1=1 THEN LOCATE fx1,fy1:PRINT "+
    "                                       [5AA8]
540 IF f2=1 THEN LOCATE fx2,fy2:PRINT "+
    "                                       [D5B0]
550 IF f3=1 THEN LOCATE fx3,fy3:PRINT "+
    "                                       [C8B8]
560 IF f4=1 THEN LOCATE fx4,fy4:PRINT "+
    "                                       [63C0]
570 tas=0:tza=0                             [74CE]
580 ged=0                                   [65B4]
590 IF bil=1 THEN PAPER 1:LOCATE xpo,ypo
    :PRINT " "                              [9B04]
600 frt=frt+1:IF frt>=frz THEN fra=fra+1
    :frt=0:frz=(500+INT(RND*1300)):IF fr
    a>1 THEN fra=0                          [80C8]
610 IF fra=1 AND bil=1 THEN PAPER 1:PEN
    2:LOCATE 38,3:PRINT "e" ELSE IF bil=
    1 THEN LOCATE 38,3:PAPER 1:PRINT " "
                                            [16E8]
620 fuz=fuz+((3-bil)*spe*2):IF fuz>=2000
     THEN fue=fue-1:MOVE fue,30:DRAW fue
    ,16,3:fuz=0:IF fue=288 THEN GOTO 419
    0                                       [E114]
630 IF xpo=38 AND ypo=3 AND fra=1 THEN G
    OTO 4420                                [721C]
640 IF bil=2 THEN GOTO 840                  [C0E6]
650 IF spe>0 THEN bew=bew+spe:IF bew>=10
    00 THEN 680                             [6756]
660 PAPER 1:PEN 2:LOCATE xpo,ypo:PRINT "
    d"                                      [7EE0]
670 GOTO 840                                [8364]
680 IF cou=1 THEN xpo=xpo-1:ypo=ypo+1       [73BA]
690 IF cou=2 THEN ypo=ypo+1                 [C1B8]
700 IF cou=3 THEN xpo=xpo+1:ypo=ypo+1       [75AC]
710 IF cou=4 THEN xpo=xpo-1                 [07AE]
720 IF cou=6 THEN xpo=xpo+1                 [8AB0]
730 IF cou=7 THEN xpo=xpo-1:ypo=ypo-1       [5EC2]
740 IF cou=8 THEN ypo=ypo-1                 [EEC0]
750 IF cou=9 THEN xpo=xpo+1:ypo=ypo-1       [80C6]
760 IF XPO<17 OR YPO>13 THEN 4070           [1478]
770 PAPER 2:PEN 3:LOCATE 34,23:PRINT xpo
    :LOCATE 34,24:PRINT ypo                 [7114]
780 IF xpo>39 OR ypo<2 THEN z=0:GOTO 455
    0                                       [0B9A]
790 bew=0:PAPER 2:PEN 3:LOCATE 34,23:PRI
    NT xpo:LOCATE 34,24:PRINT ypo           [DBE2]
800 IF xpo=fx3 AND ypo=fy3 AND f3=1 THEN
     pl=ens3:sss=3:GOTO 2230                [9B7C]
810 IF xpo=fx2 AND ypo=fy2 AND f2=1 THEN
     pl=ens2:sss=2:GOTO 2230                [2074]
820 IF xpo=fx1 AND ypo=fy1 AND f1=1 THEN
     pl=ens1:sss=1:GOTO 2230                [CD6C]
830 IF xpo=fx4 AND ypo=fy4 AND f4=1 THEN
     pl=ens4:sss=4:GOTO 2230                [628C]
840 IF INKEY(0)<>-1 AND spe<40 THEN spe=
    spe+1:PAPER 3:PEN 2:LOCATE 24,22:PRI
    NT spe                                  [7304]
850 IF INKEY(2)<>-1 AND spe>1 THEN spe=s
    pe-1:PAPER 3:PEN 2:LOCATE 24,22:PRIN
    T spe                                   [95AC]
860 j=JOY(0):ON j GOSUB 910,880,960,890,
    940,920,960,900,950,930                 [6D52]
870 GOTO 960                                [8F6E]
880 cou$="SOUTH{5 SPACE}":cou=2:ged=1:ku
    r=7:RETURN                              [D482]
890 cou$="WEST{6 SPACE}":cou=4:ged=1:kur
    =1:RETURN                               [F61C]
900 cou$="EAST{6 SPACE}":cou=6:ged=1:kur
    =5:RETURN                               [08EC]
910 cou$="NORTH{5 SPACE}":cou=8:ged=1:ku
    r=3:RETURN                              [FF6A]
920 cou$="SOUTH/WEST":cou=1:ged=1:kur=8:
    RETURN                                  [431C]
930 cou$="SOUTH/EAST":cou=3:ged=1:kur=6:
    RETURN                                  [28F2]
940 cou$="NORTH/WEST":cou=7:ged=1:kur=2:
    RETURN                                  [AD10]
950 cou$="NORTH/EAST":cou=9:ged=1:kur=4:
    RETURN                                  [6AEE]
960 IF ged=1 THEN LOCATE 2,22:PAPER 3:PE
    N 2:PRINT cou$                          [0AB6]
970 IF INKEY(23)<>-1 AND bil=2 THEN bil=
    1:xpo=xpo+1:PAPER 2:PEN 3:LOCATE 34,
    23:PRINT xpo:GOTO 460                   [4F68]
980 IF INKEY(79)<>-1 THEN GOSUB 5120        [A5C0]
990 IF bil=1 THEN 580                       [5844]
1000 glu=0                                  [592C]
1010 zu=INT(RND*2+1):IF zu=1 THEN plu=-1
      ELSE plu=1                            [EB5E]
1020 IF INT(RND*20*ens)=5 THEN tar=tar+p
     lu:glu=1:IF tar<1 THEN tar=8 ELSE I
```

```
     F tar>8 THEN tar=1                     [730E]
1030 IF glu THEN ON tar GOSUB 2270,2280,
     2290,2300,2310,2320,2330,2340          [0312]
1040 IF glu THEN PAPER 3:PEN 0:LOCATE 2,
     19:PRINT tar$                          [595E]
1050 IF kur<>tar THEN frc=frc-spe:IF frc
     <=-700 THEN 1080                       [15B0]
1060 IF spe<=((5-ens)*8) THEN frc=frc-(4
     0-spe):IF frc<=-700 THEN 1080          [C22A]
1070 GOTO 1290                              [281A]
1080 ran=ran+1:frc=0:PAPER 3:PEN 0:LOCAT
     E 7,16:PRINT ran:IF ran>20 THEN xpo
     =xpo+1:ran=0:GOTO 460                  [1DE2]
1090 GOTO 1290                              [861E]
1100 IF ran>3 OR kur<>tar THEN 1280         [42E2]
1110 IF INT(RND*50)>(ens*3) THEN 1280       [325A]
1120 v=INT(RND*2+1)                         [3DDE]
1130 PRINT CHR$(22):PAPER 2:PEN 1:LOCATE
      k(v),11:PRINT CHR$(145):PRINT CHR$
     (22)CHR$(0)                            [CEFC]
1140 SOUND 4,0,10,5,0,0,16                  [78B0]
1150 LOCATE k(v),11:PEN 0:PRINT MID$(u$(
     7),k(v)-18,1)                          [2F28]
1160 IF INT(RND*30)>5 THEN 1280             [92E2]
1170 BORDER 26,0                            [EB16]
1180 FOR j=7 TO 1 STEP -0.5:SOUND 2,0,10
     ,j,0,0,7:NEXT                          [988E]
1190 BORDER 2                               [FCF6]
1200 dam=dam+ens:LOCATE 8,24:PAPER 3:PEN
      2:PRINT dam                           [87BC]
1210 IF dam>60 THEN PAPER 3:PEN 1:LOCATE
      14,22:PRINT CHR$(231);CHR$(231);CH
     R$(231)                                [B75A]
1220 IF INT(RND*25)=2 AND tur>0 THEN tur
     =tur-1:PAPER 0:PEN 3:LOCATE 11,4+tu
     r:PRINT "-"                            [F1E8]
1230 IF INT(RND*25)=2 AND tub>0 THEN tub
     =tub-1:PAPER 0:PEN 3:LOCATE 11,6+tu
     b:PRINT "-"                            [CE6E]
1240 IF dam<70 THEN 1280                    [D64A]
1250 INK 1,24:PAPER 1:CLS                   [9398]
1260 FOR s=7 TO 1 STEP -0.2:SOUND 7,0,12
     ,s,0,0,20:NEXT                         [A20E]
1270 z=1:GOTO 4550                          [7C66]
1280 GOTO 1410                              [B114]
1290 ged=0:sch=0                            [A5DC]
1300 IF tar=kur AND spe>((5-ens)*8) THEN
      frc=frc+spe:IF frc>=700 THEN ran=r
     an-1:frc=0:ged=1:IF ran<1 THEN ran=
     1                                      [9FE8]
1310 IF ged=0 THEN 1330                     [0ECE]
1320 PAPER 3:PEN 0:LOCATE 7,16:PRINT ran
                                            [ED44]
1330 IF kur<>tar OR ran>12 THEN PAPER 2:
     LOCATE 23,11:PRINT STRING$(9," "):L
     OCATE 25,10:PRINT STRING$(5," "):GO
     TO 1390                                [4008]
1340 PAPER 2:PEN 0:IF ran>12 THEN LOCATE
      26,11:PRINT " "                       [E8E4]
1350 IF ran<4 THEN LOCATE 19,11:PRINT u$
     (7):LOCATE 19,10:PRINT u$(8):u=8       [FB8C]
1360 IF ran<7 AND ran>3 THEN LOCATE 19,1
     1:PRINT u$(5):LOCATE 19,10:PRINT u$
     (6):u=6                                [2612]
1370 IF ran<10 AND ran>6 THEN LOCATE 19,
     11:PRINT u$(3):LOCATE 19,10:PRINT u
     $(4):u=4                               [D462]
1380 IF ran<13 AND ran>9 THEN LOCATE 19,
     11:PRINT u$(1):LOCATE 19,10:PRINT u
     $(2):u=2                               [2864]
1390 REM                                    [01A2]
1400 GOTO 1100                              [9E00]
1410 IF tur=0 THEN 1550                     [2C2E]
1420 IF JOY(0)=16 AND bul>16 AND tur>0 T
     HEN GOTO 1440                          [E12A]
1430 GOTO 1550                              [3A18]
1440 a=u:tre=0                              [B8DC]
1450 SOUND 4,0,7,7,0,0,5                    [0604]
1460 sx=INT(RND*16+19):sy=INT(RND*6+6)      [B23A]
1470 PRINT CHR$(22):PAPER 2:PEN 1:LOCATE
      sx,sy:PRINT CHR$(140):PRINT CHR$(2
     2)CHR$(0)                              [1C86]
1480 IF ran>12 OR sy<10 OR tar<>kur THEN
      s$=" ":GOTO 1530                      [E0BE]
1490 IF sy=11 THEN a=a-1                    [88E8]
1500 s$=MID$(u$(a),sx-18,1)                 [29BC]
1510 IF s$<>CHR$(32) AND ran<4 THEN tre=
     1:trz=trz-1                            [30BE]
1520 IF tre=1 THEN FOR i=7 TO 1 STEP -0.
     5:SOUND 2,0,10,i,0,0,15:NEXT           [3492]
1530 PEN 0:LOCATE sx,sy:PRINT s$            [EA5E]
1540 buc=buc+1:IF buc=3 THEN MOVE bul,22
     4:DRAW bul,238,3:bul=bul-1:buc=0       [210C]
1550 IF tub=0 THEN 1650                     [011A]
1560 IF INKEY$=" " AND tas=0 AND tor>1 T
     HEN 1590                               [1E7A]
1570 IF tas=0 THEN 1710                     [DB12]
1580 GOTO 1650                              [6726]
1590 REM                                    [49A6]
1600 PAPER 2:PEN 0                          [2C3C]
1610 LOCATE 14,18:PRINT "{3 SPACE}- TORP
     EDO LAEUFT -"                          [6FBA]
1620 FOR i=20 TO 1 STEP-1:SOUND 2,0,3,3,
```




```
     0,0,i:NEXT                                  [0C76]
1630 LOCATE tor,14:PAPER 3:PRINT " ":tor
     =tor-1                                      [C4EE]
1640 tas=1                                       [9642]
1650 IF tas=1 THEN tza=tza+1:IF tza>=49
     AND ran<4 AND tar=kur THEN 1680             [B804]
1660 IF tza=50 THEN tas=0:tza=0:LOCATE 1
     4,18:PAPER 2:PRINT STRING$(25," ")          [CFC2]
1670 GOTO 1710                                   [3220]
1680 PAPER 2:PEN 1:LOCATE 14,18:PRINT "T
     REFFER UNTER WASSERLINIE":FOR i=7 T
     O 1 STEP -0.2:SOUND 1,0,10,i,0,0,20
     :NEXT                                       [AB78]
1690 LOCATE 14,18:PRINT STRING$(25," ")          [44D8]
1700 trz=trz-4:tas=0:tza=0:LOCATE 14,18:
     PAPER 2:PRINT STRING$(25," ")               [43CA]
1710 REM                                         [419A]
1720 IF trz<=0 THEN tas=0:tza=0:LOCATE 1
     4,18:PAPER 2:PRINT STRING$(25," "):
     GOTO 4640                                   [2CAE]
1730 GOTO 580                                    [78C2]
1740 REM ZERSTOERER                              [300A]
1750 u$(1)=STRING$(7," ")+CHR$(139)+STRI
     NG$(8," ")                                  [E462]
1760 u$(2)=STRING$(16," ")                       [7714]
1770 u$(3)=STRING$(6," ")+CHR$(136)+CHR$
     (137)+CHR$(138)+STRING$(7," ")              [03C2]
1780 u$(4)=u$(2)                                 [3DCE]
1790 u$(5)=STRING$(5," ")+CHR$(130)+CHR$
     (131)+CHR$(132)+CHR$(133)+CHR$(134)
     +STRING$(6," ")                             [D0F4]
1800 u$(6)=STRING$(7," ")+CHR$(135)+STRI
     NG$(8," ")                                  [F05C]
1810 u$(7)=STRING$(4," ")+"xyz{|}"+CHR$(
     126)+STRING$(5," ")                         [53F0]
1820 u$(8)=STRING$(6," ")+CHR$(127)+CHR$
     (128)+CHR$(129)+STRING$(7," ")              [8EC4]
1830 ens=3:k(1)=24:k(2)=29:trz=20:tar$="
     EAST{6 SPACE}":tar=5                        [E4F6]
1840 GOTO 2250                                   [6A1E]
1850 REM FREGATTE                                [8688]
1860 u$(1)="{8 SPACE}"+CHR$(175)+"{7 SPA
     CE}"                                        [1EE8]
1870 u$(2)=STRING$(16," ")                       [9E18]
1880 u$(3)="{8 SPACE}"+CHR$(172)+CHR$(17
     3)+CHR$(174)+"{5 SPACE}"                    [EBCC]
1890 u$(4)=u$(2)                                 [7CD2]
1900 u$(5)="{6 SPACE}"+CHR$(166)+CHR$(16
     7)+CHR$(168)+CHR$(169)+CHR$(170)+"{
     5 SPACE}"                                   [A8B8]
1910 u$(6)="{8 SPACE}"+CHR$(171)+"{6 SPA
     CE}"                                        [58A2]
1920 u$(7)="{5 SPACE}w"+CHR$(159)+CHR$(1
     60)+CHR$(161)+CHR$(162)+CHR$(163)+"
     {5 SPACE}"                                  [F14C]
1930 u$(8)="{7 SPACE}"+CHR$(164)+CHR$(16
     5)+"{7 SPACE}"                              [7EE0]
1940 ens=2:k(1)=25:k(2)=29:trz=10:tar$="
     NORTH/WEST":tar=2                           [DB12]
1950 GOTO 2250                                   [8322]
1960 REM KREUZER                                 [5F38]
1970 u$(1)="{8 SPACE}"+CHR$(192)+CHR$(19
     3)+"{7 SPACE}"                              [C91E]
1980 u$(2)=STRING$(16," ")                       [791C]
1990 u$(3)="{7 SPACE}"+CHR$(189)+CHR$(19
     0)+CHR$(191)+"{7 SPACE}"                    [541C]
2000 u$(4)=u$(2)                                 [84B2]
2010 u$(5)="{5 SPACE}"+CHR$(197)+CHR$(19
     8)+CHR$(199)+CHR$(200)+CHR$(201)+CH
     R$(202)+"{5 SPACE}"                         [7F7E]
2020 u$(6)="{6 SPACE}"+CHR$(194)+CHR$(19
     5)+CHR$(196)+"{8 SPACE}"                    [6510]
2030 u$(7)="{4 SPACE}"+CHR$(176)+CHR$(17
     7)+CHR$(178)+CHR$(179)+CHR$(180)+CH
     R$(181)+CHR$(182)+CHR$(183)+CHR$(18
     4)+"{4 SPACE}"                              [1CC0]
2040 u$(8)="{6 SPACE}"+CHR$(185)+CHR$(18
     6)+CHR$(187)+CHR$(188)+"{7 SPACE}"          [E814]
2050 ens=4:k(1)=24:k(2)=30:trz=25:tar$="
     SOUTH/EAST":tar=6                           [2CEE]
2060 GOTO 2250                                   [8F14]
2070 REM SCHNELLBOOT                             [B754]
2080 u$(1)="{7 SPACE}"+CHR$(146)+"{8 SPA
     CE}"                                        [52DA]
2090 u$(2)=STRING$(16," ")                       [CD0E]
2100 u$(3)="{6 SPACE}"+CHR$(147)+CHR$(14
     8)+"{8 SPACE}"                              [07C6]
2110 u$(4)=u$(2)                                 [97B6]
2120 u$(5)="{5 SPACE}"+CHR$(150)+CHR$(15
     1)+CHR$(152)+CHR$(153)+"{7 SPACE}"          [ED8C]
2130 u$(6)="{7 SPACE}"+CHR$(149)+"{8 SPA
     CE}"                                        [77E2]
2140 u$(7)="{5 SPACE}w"+CHR$(155)+CHR$(1
     56)+CHR$(157)+CHR$(158)+"{6 SPACE}"
                                                 [C36A]
2150 u$(8)="{7 SPACE}"+CHR$(154)+"{8 SPA
     CE}"                                        [9CE2]
2160 ens=1:k(1)=25:k(2)=29:trz=5:tar$="W
     EST{6 SPACE}":tar=1                         [11B8]
2170 GOTO 2250                                   [5A18]
2180 fx1=INT(RND*19+17):fy1=INT(RND*3+3)
     :f1=1:ens1=INT(RND*4+1)                     [1BCE]
2190 fx2=INT(RND*19+17):fy2=INT(RND*3+6)
     :f2=1:ens2=INT(RND*4+1)                     [51DE]
2200 fx3=INT(RND*19+17):fy3=INT(RND*3+9)
     :f3=1:ens3=INT(RND*4+1)                     [18DC]
2210 fx4=INT(RND*2+36):fy4=INT(RND*9+4):
     f4=1:ens4=INT(RND*4+1)                      [497A]
2220 RETURN                                      [B38C]
2230 bil=2:ran=INT(RND*6+12):tas=0               [5484]
2240 ON pl GOTO 1740,1850,2070,1960              [BD62]
2250 PAPER 3:PEN 0:LOCATE 7,16:PRINT ran
     :LOCATE 2,19:PRINT tar$                     [26A2]
2260 CLS #1:CLS #4:GOTO 580                      [C406]
2270 tar$="WEST{6 SPACE}":RETURN                 [D5E8]
2280 tar$="NORTH/WEST":RETURN                    [FFDE]
2290 tar$="NORTH{5 SPACE}":RETURN                [1A3C]
2300 tar$="NORTH/EAST":RETURN                    [10A4]
2310 tar$="EAST{6 SPACE}":RETURN                 [1AB2]
2320 tar$="SOUTH/EAST":RETURN                    [52B8]
2330 tar$="SOUTH{5 SPACE}":RETURN                [C342]
2340 tar$="SOUTH/WEST":RETURN                    [68E8]
2350 IF z=1 THEN 2420                            [3370]
2360 des$(1)="SIE HABEN DESERTIERT,INDEM
      SIE VERSUCHT "                             [042C]
2370 des$(2)="HABEN,DAS IHNEN ZUGETEILTE
      EINSATZGEBIET"                             [E5A8]
2380 des$(3)="ZU VERLASSEN.MARINEFLIEGER
      HABEN SIE ZUR"                             [CBB8]
2390 des$(4)="RUECKKEHR ZUM STUETZPUNKT
     GEZWUNGEN.SIE "                             [717C]
2400 des$(5)="WERDEN IHRES DIENSTGRADES
     ENTHOBEN UND{2 SPACE}"                      [6908]
2410 des$(6)="VOR EIN KRIEGSGERICHT GEST
     ELLT. -ENDE-{2 SPACE}":RETURN               [5F9E]
2420 des$(1)="DIE BESCHAEDIGUNG IHRES SC
     HIFFES WAR SO "                             [D0B0]
2430 des$(2)="HOCH,DASS ALLE SYSTEME ZUS
     AMMENGEBROCHEN"                             [48D6]
2440 des$(3)="SIND UND DAS SCHIFF EXPLOD
     IERT IST.DURCH"                             [5C46]
2450 des$(4)="UNUEBERLEGTES HANDELN HABE
     N SIE EIN KOM-"                             [F4E8]
2460 des$(5)="PLETTES SCNELLBOOT UND SEI
     NE BESATZUNG{2 SPACE}"                      [AB6E]
2470 des$(6)="AUF DEM GEWISSEN. -ENDE-{1
     6 SPACE}":RETURN                            [7014]
2480 des$(1)="DER BETRIEBSSTOFF IST IHNE
     N RESTLOS AUS-"                             [3684]
2490 des$(2)="GEGANGEN.SIE MUSSTEN DAS B
     OOT VERLASSEN "                             [AA58]
2500 des$(3)="UND ES SPRENGEN,DAMIT ES D
     EM FEIND NICHT"                             [D3A4]
2510 des$(4)="IN DIE HAENDE FAELLT.LEIDE
     R WURDEN SIE{2 SPACE}"                      [BC1A]
2520 des$(5)="UND IHRE BESATZUNG VON EIN
     EM FEINDLICHEN"                             [9B58]
2530 des$(6)="ZERSTOERER AUFGEFISCHT. -E
     NDE-{10 SPACE}":RETURN                      [AD9C]
2540 des$(1)="SIE HABEN IHRE MISSION VOR
     ZEITIG ABGE-{2 SPACE}"                      [406C]
2550 des$(2)="BROCHEN UND SIND ZUM STUET
     ZPUNKT ZU-{4 SPACE}"                        [FBBE]
2560 des$(3)="RUECKGEKEHRT.SIE KONNTEN I
     HREN AUFTRAG{2 SPACE}"                      [6144]
2570 des$(4)="NICHT ERFUELLEN.VOR IHREM
     NAECHSTEN EIN-"                             [4050]
2580 des$(5)="SATZ SOLLTEN SIE IHRE ANGR
     IFFSTAKTIK{4 SPACE}"                        [68B4]
2590 des$(6)="UEBERPRUEFEN. -ENDE-{20 SP
     ACE}":RETURN                                [A07C]
2600 des$(1)="SIE SIND DER FEINDLICHEN K
     UESTE ZU NAHE "                             [3A94]
2610 des$(2)="GEKOMMEN.EINE GESCHUETZSTE
     LLUNG HAT SIE "                             [A39A]
2620 des$(3)="ENTDECKT UND VERSENKT.DURC
     H SCHLECHTES{2 SPACE}"                      [2348]
2630 des$(4)="NAVIEGIEREN HABEN SIE IHR
     SCHNELLBOOT{3 SPACE}"                       [1B7A]
2640 des$(5)="LEICHTFERTIG ZERSTOERT.ZUM
      GLUECK KONNTE"                             [E866]
2650 des$(6)="SICH DIE BESATZUNG RETTEN.
      -ENDE-{7 SPACE}":RETURN                    [C0E2]
2660 RETURN                                      [AF9C]
2670 SYMBOL AFTER 99                             [D8D2]
2680 SYMBOL 100,0,0,32,58,255,127                [69CE]
2690 SYMBOL 101,4,22,86,255,127,127              [B8AE]
2700 SYMBOL 102,0,4,6,255,255,6,4                [B4C4]
2710 SYMBOL 103,0,0,0,60,90,219,255,60           [EFA2]
2720 SYMBOL 104,0,0,0,24,126,124,60              [EE6C]
2730 SYMBOL 105,0,32,96,96,96,96,96,96           [E40E]
2740 SYMBOL 106,0,0,4,4,20,20,212,84             [23CE]
2750 SYMBOL 107,0,0,0,0,224,224,232,232          [03F4]
2760 SYMBOL 108,32,32,160,224,192,192,19
     2,192                                       [248E]
2770 SYMBOL 109,96,96,96,96,96,96,248,24
     8                                           [536C]
2780 SYMBOL 110,255,170,127,63,39,63,63,
     63                                          [156E]
2790 SYMBOL 111,248,168,252,248,255,255,
     251,255                                     [6976]
2800 SYMBOL 112,0,96,96,64,88,88,80,80           [96F8]
2810 SYMBOL 113,192,192,192,192,192,192,
     192,192                                     [4B6C]
```

Listing 2. Schnellbootkommandant (Fortsetzung)

Ergänzen Sie jetzt Ihre HAPPY-COMPUTER-Sammlung

Schaffen Sie sich ein interessantes Nachschlagewerk und gleichzeitig ein wertvolles Archiv!



Kennen Sie alle »Happy Computer«-Ausgaben von 1985? Suchen Sie einen ganz bestimmten Testbericht? Oder haben Sie einen Teil eines interessanten Kurses versäumt? Suchen Sie nach einer speziellen Anwendung?

Damit Sie jetzt fehlende Hefte mit »Ihrem« Artikel nachbestellen können, finden Sie auf diesen Seiten eine Zusammenstellung aller wesentlichen Artikel der Ausgaben 01 bis 06 und 08 bis 12/85.

Und so kommen Sie schnell an die noch lieferbaren Ausgaben: Prüfen Sie, welche Ausgabe in Ihrer Sammlung noch fehlt, oder welches Thema Sie interessiert. Tragen Sie die Nummer dieser Ausgabe und das Erscheinungsjahr (z.B. 2/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein. Die ausgefüllte Zahlkarte einfach heraustrennen und Rechnungsbetrag beim nächsten Postamt einzahlen. Ihre Bestellung wird nach Zahlungseingang umgehend zur Auslieferung gebracht.

Auch die bisher erschienenen Sonderhefte können Sie jetzt direkt bestellen:

SONDERHEFT 01/84: SINCLAIR
Unentbehrliche Informationen zu den Sinclair Computern ZX81 und Spectrum.

SONDERHEFT 01/85: SPECTRUM
Anwendungsbezogene Listings und Tips&Tricks für alle Spectrum-Fans.

SONDERHEFT 02/85: SCHNEIDER 1
Eine Fülle wertvoller Beiträge und Listings für alle Schneider-Anwender.

SONDERHEFT 03/85: SPIELE
Ein Super-Nachschlagewerk für alle Spiele-Fans mit 100 Spielen im Test und großer Marktübersicht.

SONDERHEFT 01/86: SCHNEIDER 2
Noch mehr Tips und Tricks für Einsteiger und Fortgeschrittene mit vielen interessanten Programm-Listings.

SONDERHEFT 02/86: ATARI 1
Besonders 800 XL- und 130 XE-Fans erwarten jede Menge Anwendungs- und Spiele-Listings sowie Informationen.

SONDERHEFT 03/86: 68000er
Umfassende Informationen zur neuen Computer-Generation und eine große Vergleichstabelle, die im Detail über alle 68000er informiert.

Tragen Sie die Nummer des gewünschten Sonderheftes (z.B. 03/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein.

Am besten gleich mitbestellen: Die Happy Computer-Sammelbox

Für alle Leser, die »Happy Computer« regelmäßig kaufen, sammeln oder im Abonnement beziehen, gibt es ein interessantes Service-Angebot: die Happy Computer-Sammelbox!

Mit dieser Sammelbox bringen Sie nicht nur Ordnung in Ihre wertvollen Hefte, sondern schaffen sich gleichzeitig ein interessantes und attraktives Nachschlagewerk.

Übrigens: Die Sammelbox ist nicht nur ein praktisches Aufbewahrungsmittel: Sie eignet sich auch hervorragend als Geschenk für Freunde und Bekannte zu vielen Anlässen.



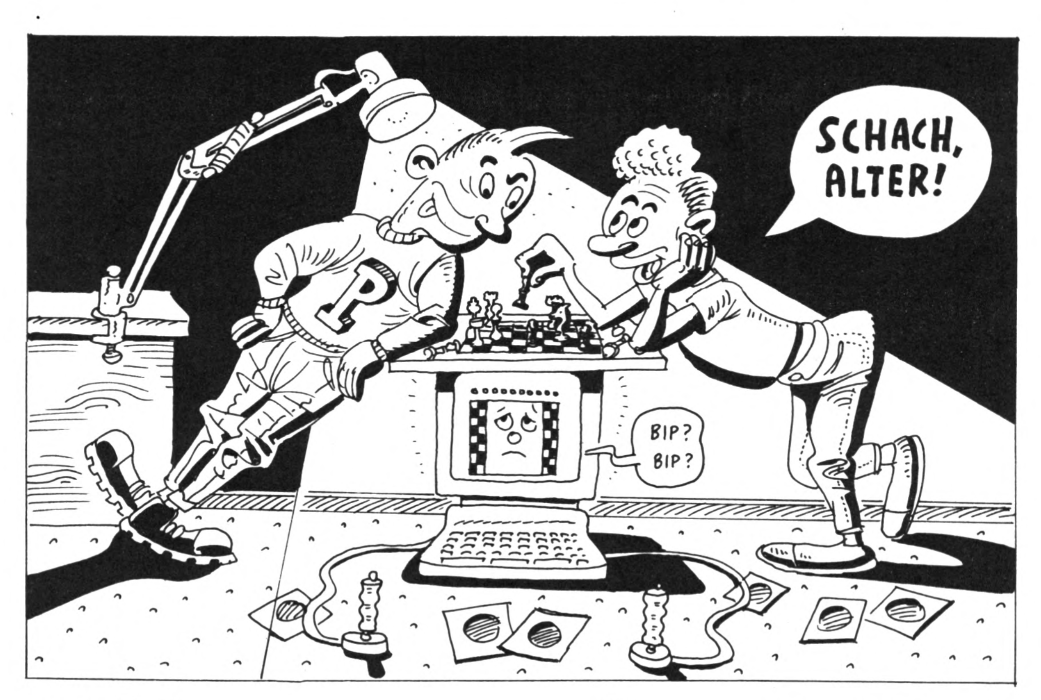
```
2820 SYMBOL 114,0,2,52,56,56,126,60,60          [97B8]
2830 SYMBOL 115,255,142,255,255,255,255,
     255,255                                    [826A]
2840 SYMBOL 116,255,143,255,255,255,254,
     252,248                                    [1D6C]
2850 SYMBOL 117,32,22,14,63,158,222,255,
     255                                        [F0C2]
2860 SYMBOL 118,127,62,31,15,7,3,1              [5236]
2870 SYMBOL 119,0,0,0,255,127,63,31,15          [A6B6]
2880 SYMBOL 120,0,0,224,255,127,31,7,1          [E3AA]
2890 SYMBOL 121,0,13,29,221,255,255,255,
     255                                        [5FC2]
2900 SYMBOL 122,172,255,255,255,255,255,
     255,255                                    [4B68]
2910 SYMBOL 123,253,255,255,255,255,255,
     255,255                                    [336C]
2920 SYMBOL 124,25,31,255,255,255,255,25
     5,255                                      [F19A]
2930 SYMBOL 125,200,241,253,255,255,255,
     255,255                                    [3356]
2940 SYMBOL 126,0,128,192,192,255,254,25
     2,248                                      [149E]
2950 SYMBOL 127,0,0,0,1,28,8,14,46              [7B24]
2960 SYMBOL 128,16,208,240,193,193,193,1
     93,249                                     [3418]
2970 SYMBOL 129,0,0,0,0,128,48,56,16            [85FA]
2980 SYMBOL 130,0,0,0,192,254,127,31,7          [A5B2]
2990 SYMBOL 131,12,8,95,223,223,255,255,
     255                                        [5AD4]
3000 SYMBOL 132,194,242,250,255,255,255,
     255,255                                    [F650]
3010 SYMBOL 133,192,64,108,255,255,255,2
     55,255                                     [21F8]
3020 SYMBOL 134,0,0,32,48,176,255,254,25
     2                                          [4BE4]
3030 SYMBOL 135,0,0,0,0,192,224,194,194         [D50A]
3040 SYMBOL 136,0,0,0,0,5,237,127,31            [93CC]
3050 SYMBOL 137,32,48,36,181,253,255,255
     ,255                                       [3636]
3060 SYMBOL 138,0,0,0,0,232,236,255,254         [1E08]
3070 SYMBOL 139,0,0,0,0,32,40,124,255           [2132]
3080 SYMBOL 140,56,68,82,165,139,161,74,
     36                                         [B57C]
3090 SYMBOL 141,0,0,0,12,30,62,126,254          [A894]
3100 SYMBOL 142,0,0,0,0,0,32,112,248            [86B0]
3110 SYMBOL 143,7,15,15,15,63,127,255,25
     5                                          [26EE]
3120 SYMBOL 144,252,252,252,252,252,248,
     240,224                                    [3E3C]
3130 SYMBOL 145,112,112,112                     [606E]
3140 SYMBOL 146,0,0,0,0,0,8,222,127             [5A62]
3150 SYMBOL 147,0,0,0,0,2,246,127,63            [7AD8]
3160 SYMBOL 148,0,32,32,96,248,250,255,2
     55                                         [3766]
3170 SYMBOL 149,0,0,0,0,32,32,32,96             [2E7A]
3180 SYMBOL 150,0,0,0,0,254,127,63,31           [9E34]
3190 SYMBOL 151,0,3,35,103,119,255,255,2
     55                                         [2850]
3200 SYMBOL 152,32,160,249,255,255,255,2
     55,255                                     [A5F4]
3210 SYMBOL 153,0,192,240,244,246,255,25
     5,255                                      [9382]
3220 SYMBOL 154,0,0,3,1,1,7,3,3                 [E4D2]
3230 SYMBOL 155,0,4,12,142,255,255,255,2
     55                                         [6D4E]
3240 SYMBOL 156,123,127,255,255,255,255,
     255,255                                    [4766]
3250 SYMBOL 157,3,199,255,255,255,255,25
     5,255                                      [B6B6]
3260 SYMBOL 158,128,224,244,246,255,255,
     255,255                                    [D970]
3270 SYMBOL 159,7,15,15,230,255,255,255,
     255                                        [A8D2]
3280 SYMBOL 160,31,63,127,127,255,255,25
     5,255                                      [D596]
3290 SYMBOL 161,236,238,238,238,255,255,
     255,255                                    [F67C]
3300 SYMBOL 162,1,1,253,124,255,255,255,
     255                                        [7AB0]
3310 SYMBOL 163,192,224,224,196,255,254,
     252,248                                    [AE62]
3320 SYMBOL 164,0,1,0,0,0,6,4,7                 [8CD6]
3330 SYMBOL 165,32,32,224,192,192,192,19
     2,200                                      [4A7E]
3340 SYMBOL 166,0,0,0,0,127,63,31,15            [46D4]
3350 SYMBOL 167,0,0,25,59,27,255,255,255
                                                [D4A2]
3360 SYMBOL 168,56,253,253,253,253,255,2
     55,255                                     [A610]
3370 SYMBOL 169,0,0,128,192,223,255,255,
     255                                        [29CA]
3380 SYMBOL 170,0,0,48,56,48,255,254,252
                                                [0D9E]
3390 SYMBOL 171,0,0,0,4,12,8,8,8                [585A]
3400 SYMBOL 172,0,0,0,0,2,246,127,63            [C4D0]
3410 SYMBOL 173,8,8,8,58,123,251,255,255
                                                [46A8]
3420 SYMBOL 174,0,0,0,0,4,118,255,254           [0340]
3430 SYMBOL 175,0,0,0,0,0,16,52,255             [646E]
3440 SYMBOL 176,0,0,0,240,255,127,63,31         [E510]
3450 SYMBOL 177,0,3,119,247,255,255,255,
     255                                        [CCD8]
3460 SYMBOL 178,15,191,191,191,255,255,2
     55,255                                     [DA14]
3470 SYMBOL 179,255,255,255,255,255,255,
     255,255                                    [818A]
3480 SYMBOL 180,128,248,255,255,255,255,
     255,255                                    [DC7E]
3490 SYMBOL 181,189,189,255,255,255,255,
     255,255                                    [F498]
3500 SYMBOL 182,128,238,255,255,255,255,
     255,255                                    [0072]
3510 SYMBOL 183,0,0,0,112,120,255,255,25
     5                                          [17CC]
3520 SYMBOL 184,0,0,0,0,0,255,254,252           [BD3A]
3530 SYMBOL 185,0,0,0,3,15,15,7,3               [F8B0]
3540 SYMBOL 186,16,24,48,16,215,215,215,
     215                                        [D2C2]
3550 SYMBOL 187,0,0,1,0,1,0,128,128             [5E76]
3560 SYMBOL 188,128,128,128,128,192,128,
     144,184                                    [DC7C]
3570 SYMBOL 189,0,0,0,7,3,31,255,127            [2EF6]
3580 SYMBOL 190,64,66,66,82,90,255,255,2
     55                                         [2692]
3590 SYMBOL 191,0,0,0,128,192,244,255,25
     4                                          [C5F4]
3600 SYMBOL 192,0,0,0,4,5,7,31,255              [3316]
3610 SYMBOL 193,0,0,0,0,0,160,252,255           [A032]
3620 SYMBOL 194,0,0,0,0,0,0,0,1                 [C0C0]
3630 SYMBOL 195,0,0,0,0,16,16,144,218           [8E46]
3640 SYMBOL 196,0,0,0,0,16,16,16,48             [4D88]
3650 SYMBOL 197,0,0,0,3,231,127,63,31           [874A]
3660 SYMBOL 198,1,0,51,119,119,255,255,2
     55                                         [E570]
3670 SYMBOL 199,215,215,215,255,255,255,
     255,255                                    [257A]
3680 SYMBOL 200,16,22,23,215,255,255,255
     ,255                                       [A922]
3690 SYMBOL 201,0,0,64,118,255,255,255,2
     55                                         [6E5E]
3700 SYMBOL 202,0,0,0,0,96,112,255,254          [CD9C]
3710 SYMBOL 203,0,6,15,15,6,14,6,62             [4282]
3720 SYMBOL 204,64,44,30,28,255,255,255,
     255                                        [B5CE]
3730 SYMBOL 205,126,127,255,255,255,255,
     255,255                                    [EF6A]
3740 SYMBOL 206,0,250,251,255,255,255,25
     5,255                                      [DA8E]
3750 SYMBOL 207,1,26,188,156,255,255,255
     ,254                                       [3946]
3760 SYMBOL 208,16,16,16,24,56,56,60,60         [663A]
3770 SYMBOL 209,0,0,0,0,255,30,3,1              [440A]
3780 SYMBOL 210,124,110,78,223,255,63,25
     5,255                                      [998E]
3790 SYMBOL 211,0,0,0,0,255,248,192,128         [DD1C]
3800 SYMBOL 212,1,2,3,3,7,6,12,8                [074E]
3810 SYMBOL 213,127,231,195,129                 [0526]
3820 SYMBOL 214,128,192,192,192,224,96,4
     8,16                                       [BB46]
3830 RETURN                                     [A39C]
3840 REM DATAS LANDKARTE                        [032C]
3850 DATA 228,376,234,355,232,350,226,34
     8,224,346,216,342,218,332,220,320          [1FB6]
3860 DATA 224,316,228,310,220,300,224,28
     0,228,250,224,242,222,236,226,226          [F984]
3870 DATA 230,216,232,204,234,200,240,18
     4,245,180,252,176,260,172,272,174          [169A]
3880 DATA 290,180,314,186,330,184,336,17
     8,338,172,336,158,355,168,376,174          [AC0A]
3890 DATA 390,182,412,178,430,176,440,17
     0,446,164,450,162,460,164,480,170          [0BD2]
3900 DATA 500,176,510,178,524,180,536,17
     4,545,176,555,178,570,170,595,174          [E4FE]
3910 DATA 612,170,624,176,1,1                   [CEA0]
3920 REM DATAS KUESTE                           [82A0]
3930 DATA 220,235,235,230,250,235,260,24
     5,274,238,285,227,310,232,315,237          [E9A4]
3940 DATA 325,242,337,230,340,235,349,25
     4,370,254,375,240,400,242,430,225          [08A8]
3950 DATA 450,237,462,227,470,232,478,24
     0,490,235,500,230,510,237,540,240          [74B0]
3960 DATA 550,236,570,230,593,237,610,23
     2,620,240,624,234,1,1                      [D8FC]
3970 DATA 319,20,319,12,319,30,284,20,35
     8,20,284,20,239,40,1,1                     [C64C]
3980 DATA 2,77,13,95,13                         [9668]
3990 DATA 3,77,13,95,7,106,13                   [37B8]
4000 DATA 3,77,13,95,13,112,13                  [90EA]
4010 DATA 4,77,13,95,13,112,7,123,13            [8F38]
4020 DATA 4,77,13,95,13,112,13,129,13           [96A0]
4030 DATA 2,70,60,134,7                         [983E]
4040 DATA 2,70,60,134,13                        [A99A]
4050 DATA 3,70,60,134,13,151,13                 [8D44]
4060 DATA 4,70,60,134,13,151,13,168,13          [73FE]
4070 CLS #1:CLS #4:RESTORE 3930                 [B042]
4080 PLOT 208,245,3                             [4716]
4090 READ a,b                                   [C830]
4100 IF a=1 THEN GOTO 4130                      [79E6]
4110 DRAW a,b,3                                 [5D04]
4120 GOTO 4090                                  [A31A]
4130 PAPER 2:PEN 0:LOCATE 23,9:PRINT "g"
                                                [A61A]
4140 FOR w=1 TO 500:NEXT                        [688C]
4150 PRINT CHR$(22):LOCATE 23,9:PEN 3:PR
     INT "h"                                    [C558]
4160 SOUND 1,0,8,6,0,0,5:FOR wt=1 TO 300
```




```
      :NEXT:PRINT CHR$(22)CHR$(0):LOCATE
      23,9:PEN 0:PRINT "g"                    [A206]
4170 PAPER 2:PEN 0:LOCATE 14,18:PRINT "T
      REFFER MITTSCHIFFS-SINKEN"              [3182]
4180 FOR i=7 TO 1 STEP -0.3:SOUND 1,0,10
      ,i,0,0,10:NEXT:GOTO 4330                [2898]
4190 PAPER 2:PEN 0:LOCATE 14,18:PRINT "K
      EIN BETRIEBSSTOFF MEHR !"               [8B04]
4200 FOR wt=1 TO 15:SOUND 1,478,50,6,0,1
      :NEXT                                   [A72E]
4210 LOCATE 14,18:PRINT "BESATZUNG IN DI
      E BOOTE ! "                             [5A74]
4220 FOR wt=1 TO 15:SOUND 1,478,50,6,0,1
      :NEXT                                   [BF32]
4230 FOR wt=1 TO 30:SOUND 1,0,5,0:NEXT        [396A]
4240 MODE 0:PAPER 2:CLS                       [31FE]
4250 WINDOW #1,1,20,20,25:PAPER #1,1:CLS
      #1                                      [91AC]
4260 PEN 0:LOCATE 8,19:PRINT "w"+CHR$(20
      4)+CHR$(205)+CHR$(206)+CHR$(207):LO
      CATE 10,18:PRINT CHR$(203)              [0C92]
4270 FOR wt=1 TO 500:NEXT                     [5D7C]
4280 LOCATE 10,18:PRINT CHR$(140):FOR i=
      7 TO 1 STEP -0.5:SOUND 2,0,10,i,0,0
      ,15:NEXT:LOCATE 10,18:PRINT " "         [5D42]
4290 FOR s=8 TO 12                            [7748]
4300 LOCATE s-1,19:PRINT " "+CHR$(140):F
      OR i=7 TO 1 STEP -0.5:SOUND 2,0,10,
      i,0,0,15:NEXT:c=INT(RND*30+10):FOR
      wt=1 TO c:NEXT:NEXT:LOCATE 12,19:PR
      INT " "                                 [CD04]
4310 FOR wt=1 TO 500:NEXT                     [3672]
4320 GOSUB 2480:GOTO 4560                     [1F72]
4330 FOR w=0 TO 399 STEP 5                    [461C]
4340 BORDER 1,24                              [8A18]
4350 SOUND 1,478,50,6,0,1                     [5082]
4360 FOR wa=w TO w+5                          [A374]
4370 MOVE 0,wa:DRAW 639,wa,1                  [6808]
4380 NEXT                                     [345C]
4390 SOUND 4,0,INT(RND*50),4,0,0,INT(RND
      *20)                                    [7CE8]
4400 NEXT                                     [E54E]
4410 GOSUB 2600:GOTO 4560                     [B866]
4420 CLS #1:CLS #4                            [59A6]
4430 PEN 0:PAPER 2                            [0444]
4440 LOCATE 23,9:PRINT f$(1):LOCATE 23,1
      0:PRINT f$(2):LOCATE 23,11:PRINT f$
      (3)                                     [5DE6]
4450 LOCATE 14,18:PRINT "UEBERNEHMEN BET
      RIEBSSTOFF"                             [A414]
4460 FOR wt=1 TO 1000:NEXT:FOR wt=1 TO 3
      00:SOUND 1,956,5,4:NEXT                 [BF32]
4470 LOCATE 2,11:PRINT STRING$(10," "):L
      OCATE 2,14:PRINT STRING$(10," ")        [DF48]
4480 tor=12:fue=463:bul=174                   [251E]
4490 PAPER 0:LOCATE 19,24:PRINT "{2 SPAC
      E}":PAPER 2:LOCATE 21,24:PRINT STRI
      NG$(9," ")                              [F072]
4500 FOR i=1 TO 3:SOUND 1,127,20,5:SOUND
       1,0,15,5:NEXT                          [F904]
4510 FOR wt=1 TO 1500:NEXT                    [3AD8]
4520 cou$="SOUTH":cou=2:PAPER 3:PEN 2:LO
      CATE 2,22:PRINT cou$                    [87E8]
4530 LOCATE 14,18:PAPER 2:PRINT STRING$(
      25," ")                                 [19D8]
4540 ypo=4:GOTO 460                           [BDC6]
4550 GOSUB 2350                               [E2B0]
4560 BORDER 1:MODE 1:PAPER 0:PEN 2:LOCAT
      E 1,5:PRINT "KOMMANDANT !"              [5832]
4570 FOR WT=1 TO 6:SOUND 3,0,10,0:NEXT        [2FFA]
4580 FOR i=1 TO 6:FOR s=1 TO 40               [36A2]
4590 PRINT MID$(des$(i),s,1);:IF MID$(de
      s$(i),s,1)<>CHR$(32) THEN SOUND 1,0
      ,2,7,0,0,1                              [8972]
4600 FOR wt=1 TO 60:NEXT:NEXT:NEXT            [E0FC]
4610 IF z=0 THEN INK 1,9:GOTO 5000            [3EC0]
4620 LOCATE 1,24:PRINT "<ENTER> DRUECKEN
      ":IF INKEY(18)<>-1 THEN 4900            [BEA4]
4630 GOTO 4620                                [6124]
4640 FOR i=1 TO 16                            [582C]
4650 z$=MID$(u$(u),i,1)                       [AED4]
4660 IF z$<>CHR$(32) THEN LOCATE i+16,10
      :PRINT "{3 SPACE}";CHR$(140):FOR a=
      7 TO 1 STEP -0.2:SOUND 2,0,10,a,0,0
      ,20:NEXT                                [292E]
4670 NEXT                                     [EF60]
4680 LOCATE 25,10:PRINT "{6 SPACE}"           [9652]
4690 LOCATE 23,11:PRINT "{2 SPACE}";CHR$
      (143);CHR$(144);"{6 SPACE}":LOCATE
      26,10:PRINT CHR$(142)                   [BA14]
4700 FOR wt=1 TO 200:SOUND 2,0,2,3,0,0,5
      :NEXT                                   [D400]
4710 LOCATE 26,10:PRINT " ":LOCATE 25,11
      :PRINT CHR$(141);" ":FOR wt=1 TO 10
      0:SOUND 2,0,2,2,0,0,10:NEXT:LOCATE
      25,11:PRINT " "                         [4788]
4720 FOR wt=1 TO 1000:NEXT                    [44D4]
4730 IF sss=1 THEN f1=0 ELSE IF sss=2 TH
      EN f2=0 ELSE IF sss=3 THEN f3=0 ELS
      E IF sss=4 THEN f4=0                    [C3B2]
4740 schi=schi+1:IF schi=4 THEN schi=0:G
      OTO 4760                                [167C]
4750 GOTO 460                                 [94C6]
4760 SYMBOL AFTER 255:PAPER 1:PEN 2:BORD
      ER 3:CLS                                [DFDC]
4770 LOCATE 2,2:PRINT "KOMMANDANT:"           [1C52]
4780 LOCATE 1,4:PRINT " Es ist Ihnen gel
      ungen,einen kompletten{2 SPACE}Verb
      and des Gegners zu zerstoeren."         [F0D6]
4790 PRINT " Die Admiralitaet hat Ihre B
      efoerderung{2 SPACE}beschlossen.Her
      zlichen Glueckwunsch."                  [4FB2]
4800 PRINT:PRINT " Es haben sich jedoch
      neue Schiffe zur{3 SPACE}Kuestenver
      teidigung gesammelt.Sie er-{3 SPACE
      }halten den Befehl,sich in Ihr alte
      s{5 SPACE}Einsatzgebiet zu begeben
      und diesen{5 SPACE}Verband anzugrei
      fen. -ENDE-"                            [3CB2]
4810 PEN 3:LOCATE 2,20:PRINT "< TASTE DR
      UECKEN >"                               [CA98]
4820 RESTORE 3970                             [240A]
4830 READ a,b                                 [7C34]
4840 IF a=1 THEN 4880                         [1862]
4850 SOUND 1,a,b,6                            [5890]
4860 SOUND 1,0,20,1                           [4126]
4870 GOTO 4830                                [9A36]
4880 WHILE INKEY$="":WEND                     [F134]
4890 lev=lev+1:tur=2:tub=2:fue=463:dam=0
      :tor=11:bul=174:xpo=30:ypo=2:cou=2:
      spe=1:bil=1:fuz=0:fra=1:frt=0:frz=1
      200:IF lev=8 THEN GOSUB 2670:GOTO 5
      160 ELSE GOTO 140                       [4996]
4900 BORDER 3:MODE 1:PAPER 0:INK 0,2:INK
       1,0:CLS:IF lev>8 THEN lev=8            [0960]
4910 ON lev+1 GOSUB 5030,5040,5050,5060,
      5070,5080,5090,5100,5110                [3910]
4920 WINDOW #1,15,26,7,21:PAPER #1,1:CLS
      #1                                      [57CC]
4930 PEN 3:LOCATE 14,3:PRINT "IHR DIENST
      GRAD"                                   [DB1E]
4940 INK 2,24:INK 3,13:LOCATE x,23:PRINT
       rank$                                  [8810]
4950 MOVE 0,0:DRAW 0,399,2:DRAW 639,399:
      DRAW 639,0:DRAW 0,0                     [53A2]
4960 PEN 2:PAPER 1:LOCATE 20.5,sy:PRINT
      CHR$(208):LOCATE 19.5,sy+1:PRINT CH
      R$(209)+CHR$(210)+CHR$(211):LOCATE
      19.5,sy+2:PRINT CHR$(212)+CHR$(213)
      +CHR$(214)                              [38FC]
4970 READ a:FOR z=1 TO a:READ b,c:FOR zv
      =b TO b+c                               [7E10]
4980 MOVE 224,zv:DRAW 414,zv,2:NEXT:NEXT
                                              [2C1A]
4990 PAPER 0:LOCATE 4,24:PRINT STRING$(3
      3,"_")                                  [82FC]
5000 PRINT CHR$(22):PEN 1:LOCATE 4,25:PR
      INT "FUER NEUES SPIEL <SHIFT> DRUEC
      KEN"                                    [1364]
5010 IF INKEY(21)<>-1 THEN RUN                [530A]
5020 GOTO 5010                                [6B0C]
5030 rank$="OBERLEUTNANT":sy=15:x=15:RES
      TORE 3980:RETURN                        [25AC]
5040 rank$="KAPITAENLEUTNANT":sy=14:x=13
      :RESTORE 3990:RETURN                    [B5F4]
5050 rank$="KORVETTENKAPITAEN":sy=13:x=1
      3:RESTORE 4000:RETURN                   [0D80]
5060 rank$="FREGATTENKAPITAEN":sy=13:x=1
      3:RESTORE 4010:RETURN                   [8940]
5070 rank$="KAPITAEN ZUR SEE":sy=12:x=13
      :RESTORE 4020:RETURN                    [763E]
5080 rank$="FLOTTILLENADMIRAL":sy=13:x=1
      3:RESTORE 4030:RETURN                   [785C]
5090 rank$="KONTERADMIRAL":sy=12:x=15:RE
      STORE 4040:RETURN                       [430E]
5100 rank$="VIZEADMIRAL":sy=11:x=15:REST
      ORE 4050:RETURN                         [49D4]
5110 rank$="ADMIRAL":sy=10:x=17:RESTORE
      4060:RETURN                             [8E5E]
5120 PAPER 2:PEN 0:LOCATE 14,18:PRINT "A
      UFGEBEN <J/N> ?"                        [A6A6]
5130 IF INKEY(45)<>-1 THEN GOSUB 2540:GO
      TO 4560                                 [D2CC]
5140 IF INKEY(46)<>-1 THEN LOCATE 14,18:
      PRINT STRING$(20," "):RETURN            [3B58]
5150 GOTO 5130                                [CA1A]
5160 MODE 0:PAPER 2:WINDOW #1,1,20,22,25
      :PAPER #1,1:CLS:CLS #1                  [7C50]
5170 n$="{2 SPACE}*** GRATULATION.SIE KO
      NNTEN ALLE IHNEN GESTELLTEN AUFGABE
      N ERFUELLEN UND SIND IN DER RANGLIS
      TE BIS ZUM ADMIRAL AUFGESTIEGEN ***
      {22 SPACE}SIE SOLLTEN SICH JETZT EI
      NMAL AN EINEM TAKTIKSPIEL VERSUCHEN
       !!!{19 SPACE}"                         [D16E]
5180 PEN 0:LOCATE 9,21:PRINT "w"+CHR$(20
      4)+CHR$(205)+CHR$(206)+CHR$(207):LO
      CATE 11,20:PRINT CHR$(203)              [217E]
5190 i=1                                      [078C]
5200 LOCATE 2,4:PRINT MID$(n$,i,18):FOR
      wt=1 TO 100:NEXT:i=i+1:IF i=LEN(n$)
       THEN 4900 ELSE 5200                    [4F58]
5210 END                                      [FC7E]
```

Listing 2. Schnellbootkommandant (Schluß)





Eröffnungen im Schach



Die grundlegenden Schachregeln kennen viele. Aber die Kunst der richtigen Eröffnung will gelernt sein. Lassen Sie sich von Ihrem CPC unterrichten.

Das integrierte Schachspiel wird durch eine einfache Block-Grafik dargestellt und beherrscht die üblichen Schachregeln, wie kleine und große Rochade, en-passant-Schlagen etc. Eine Einschränkung erfährt lediglich die Bauernumwandlung, da sich hier grundsätzlich der Bauer in eine Dame verwandelt. Dies stellt jedoch kaum ein Handicap dar, da dieser taktische Zug im Eröffnungsstadium sehr selten, und eine Umwandlung des Bauern in eine andere Figur als die Dame so gut wie nie vorkommt!

Das Programm belegt mit Variablen etwas über 20 KByte Speicherplatz. Wenn man pro eingegebener Variante einen Durchschnitt von 20 ganzen Zügen annimmt, erlaubt ein Durchgang gleichzeitig rund 150 Varianten. Zweckmäßigerweise legt man sich mehrere Variantenblöcke zu, die man nach Eröffnungen einteilt und getrennt auf Band oder Diskette abspeichert.

Nach dem Programmstart erscheint das Menü. Sind noch keine Varianten abgespeichert oder will man neue Varianten eingeben, so wählt man Menüpunkt 1. Die Varianteneingabe stellt den kritischen Teil des Programms dar, weil hier einige Regeln streng zu beachten sind. Die Eingabe läßt sich am besten an einem Beispiel erklären. Angenommen, Sie wollen folgende Variante der spanischen Eröffnung eintippen:

1. e2 - e4	**e7 - e5**	**5. 0 - 0**	**Lf8 - e7**
2. Sgl - f3	**Sb8 - c6**	**6. Tfl - el**	**b7 - b5**
3. Lfl - b5	**a7 - a6**	**7. La4 - b3**	**0 - 0**
4. Lb5 - a4	**Sg8 - f6**		

Die Abfrage der Züge soll mit dem vierten Zug von Schwarz beginnen. Dann wird die Variante wie folgt (beliebig in Klein- oder Großbuchstaben) eingegeben:

»e2e4«	eingeben und große ENTER-Taste drücken.
»e7e5«	»ENTER«
»g1f3«	»ENTER«
»b8c6«	»ENTER«
»f1b5«	»ENTER«
»a7a6«	»ENTER«
»xxxx«	»ENTER«
»b5a4«	»ENTER«
»g8f6«	»ENTER« (von hier ab beginnt der Computer später mit der Abfrage der Züge).
»e1g1«	»ENTER« (Rochade)

Das Drücken der »Z«-Taste beendet den Vorgang. Besonders zu beachten sind folgende Punkte:

1. Die Eingabe von »xxxx« muß in jeder Variante einmal (und nur einmal) irgendwo zwischen dem ersten und letzten eingegebenen Halbzug vorgenommen werden. »xxxx« darf also nicht an der ersten oder letzten Stelle stehen.
2. Eine Variante darf nicht mehr als 31 ganze Züge (also 62 Halbzüge) beinhalten (Eröffnungsvarianten sind in aller Regel nicht länger als 25 Züge).
3. Besonders wichtig: Bei der Eingabe von Zügen führt der Computer keine Plausibilitätskontrolle durch. Das bedeutet, daß auch sinnlose Züge (etwa ein Königssprung über zehn Felder) vom Programm kritiklos angenommen werden. Der Anwender muß also selbst darauf achten, daß seine Züge auch richtig sind.

Ist während der Eingabe ein Fehler unterlaufen, so bricht man das Programm durch zweimaliges Drücken der ESC-Taste ab. Drückt man dann die kleine ENTER-Taste auf dem Zahlenblock, gelangt man auf diese Weise wieder ins Menü und wählt die 1. Jetzt kann man die Variante neu eingeben.

Menüpunkt 2 führt in den Abfragemodus. Der Computer gibt zuerst die Stellung vor, die bei der Eingabe mit »xxxx« markiert wurde (genauer: die Stellung bis zum ersten Halbzug nach »xxxx« – die Abfrage beginnt immer mit dem zweiten Halbzug nach »xxxx«).

Sind alle Varianten durchlaufen, schließt sich die Frage nach einer Wiederholung an. Auf diese Weise kann man den Variantenkomplex so lange durchpauken, bis alles »sitzt«. Dann bietet der Computer auf die Anforderung einer Wiederholung keine Variante mehr an, weil alle richtig beantwortet sind, sondern wiederholt seine Frage. Man verneint jetzt und gelangt ins Menü zurück.

(P. Meyerbeck/ja)



```
1000 REM ================================
     ===========                         [5A48]
1010 REM ***
         ***                             [C304]
1020 REM ***   C  H  E  S  S  -  T  U  T
      O  R   ***                         [ADC8]
1030 REM ***
         ***                             [FD08]
1040 REM ***   (c) 1985 by P.Meyerbeck
         ***                             [5000]
1050 REM ***       Richard Strauss Str.
     111     ***                         [CA24]
1060 REM ***       8000 Muenchen 80
         ***                             [CD54]
1070 REM ***       Tel. (089) 986056
         ***                             [445E]
1080 REM ***
         ***                             [4A12]
1090 REM ================================
     ===========                         [255A]
1100 :                                   [C538]
1110 REM *** Deklarationen **********    [5C14]
1120 :                                   [C73C]
1130 KEY 139,"n=i-1:goto 1240"+CHR$(13)  [2E50]
1140 DEFINT a-z:DIM var$[100],r[100],ric
     htig[100]:n=0                       [681A]
1150 MODE 1:INK 0,1:INK 1,24:INK 2,10:IN
     K 3,14:BORDER 3                     [40D6]
1160 :                                   [BB44]
1170 SYMBOL 240,219,219,60,60,60,60,126,
     255                                 [70B4]
1180 SYMBOL 241,8,24,124,124,88,24,60,12
     6                                   [4FF0]
1190 SYMBOL 242,0,0,0,24,60,60,24,60     [20CE]
1200 SYMBOL 243,24,126,255,60,24,24,24,1
     26                                  [2D3C]
1210 SYMBOL 244,24,126,24,60,60,24,24,12
     6                                   [3FD4]
1220 SYMBOL 245,24,60,126,60,24,24,24,12
     6                                   [BED8]
1230 :                                   [BE40]
1240 REM --------------------            [7ADE]
1250 REM *** AUSWAHLLISTE ***            [817C]
1260 REM --------------------            [5EE2]
1270 :                                   [CA48]
1280 MODE 1:PAPER 2:PEN 0:CLS            [5CDC]
1290 LOCATE 6,5:PRINT CHR$(150);STRING$(
     29,CHR$(154));CHR$(156)             [55BA]
1300 LOCATE 6,6:PRINT CHR$(149);"{3 SPAC
     E}A U S W A H L L I S T E{3 SPACE}"
     ;CHR$(149)                          [1C16]
1310 LOCATE 6,7:PRINT CHR$(147);STRING$(
     29,CHR$(154));CHR$(153)             [2CB6]
1320 LOCATE 6,10:PRINT"[1]{3 SPACE}Varia
     nten eingeben."                     [1A10]
1330 LOCATE 6,12:PRINT"[2]{3 SPACE}Varia
     nten abfragen."                     [E20A]
1340 LOCATE 6,14:PRINT"[3]{3 SPACE}Varia
     nten abspeichern."                  [2BAE]
1350 LOCATE 6,16:PRINT"[4]{3 SPACE}Varia
     nten einlesen."                     [0854]
1360 LOCATE 6,18:PRINT"[E]{3 SPACE}Ende.
     "                                   [A93E]
1370 LOCATE 6,21:PRINT CHR$(24)+" Wahl d
     urch Tastendruck ! "+CHR$(24)       [86C8]
1380 w$=INKEY$:IF w$="" THEN 1380        [E850]
1390 ON VAL(w$) GOSUB 1470,1870,2580,264
     0                                   [92E8]
1400 IF UPPER$(w$)="E" THEN CLS:END      [71B0]
1410 GOTO 1280                           [9914]
1420 :                                   [D142]
1430 REM --------------------------      [78FC]
1440 REM *** VARIANTEN EINGEBEN ***      [D75C]
1450 REM --------------------------      [4100]
1460 :                                   [BD4A]
1470 WINDOW#0,1,19,5,21:PAPER#0,0:PEN#0,
     1:CLS#0                             [9A2E]
1480 WINDOW#1,20,40,1,21:PAPER#1,3:CLS#1
                                         [55F0]
1490 WINDOW#2,1,19,1,4:PAPER#2,0:PEN#2,1
     :CLS#2                              [14DC]
1500 WINDOW#3,1,19,22,25:PAPER#3,3:PEN#3
     ,0:CLS#3                            [ACA4]
1510 WINDOW#4,20,40,22,25:PAPER#4,2:PEN#
     4,1:CLS#4:GOSUB 2820                [9850]
1520 LOCATE#2,1,2:PRINT#2,CHR$(150);STRI
     NG$(16,CHR$(154));CHR$(156)         [1B1E]
1530 LOCATE#2,1,3:PRINT#2,CHR$(149);"VAR
     IANTENEINGABE";CHR$(149)            [3C4E]
1540 LOCATE#2,1,4:PRINT#2,CHR$(147);STRI
     NG$(16,CHR$(154));CHR$(153)         [372C]
1550 :                                   [BC4A]
1560 i=n:zg$="":s1=1:PRINT CHR$(7)       [5DF8]
1570 :                                   [CE4E]
1580 i=i+1:var$[i]="":v$="" '*** Schleif
     enanfang Varianten ***              [397C]
1590 :                                   [D852]
1600 CLS#3:LOCATE#3,2,2:PRINT#3,"Naechst
     e Variante"                         [186A]
1610 LOCATE#3,2,3:PRINT#3,"{2 SPACE}>Z<
     druecken !"                         [510C]
1620 LOCATE#4,4,3:PRINT#4,"Variante Nr."
     ;i                                  [6E62]
1630 :                                   [D648]
1640 FOR k=1 TO 2 '*** Schleifenanfang Z
     uege ***                            [3AFE]
1650 :                                   [C84C]
1660 LOCATE#4,4,2:PRINT#4,"{6 SPACE}"    [A910]
1670 LOCATE#4,4,2:INPUT#4,zg$:zg$=UPPER$
     (zg$)                               [5E1E]
1680 IF INSTR(zg$,"Z") THEN 1800 '*** Na
     echste Variante ***                 [711C]
1690 var$[i]=var$[i]+zg$:v$=var$[i]      [3130]
1700 IF zg$="XXXX" THEN s2=s1 MOD 14:k=k
     -1:GOTO 1760                        [BE06]
1710 s2=s1 MOD 14:IF s2=0 THEN CLS       [AA80]
1720 LOCATE 2,s2+2:PRINT USING "##";s1;:
     PRINT "."                           [AF58]
1730 IF k=1 THEN LOCATE 6,s2+2 ELSE LOCA
     TE 13,s2+2                          [28E2]
1740 li$=LEFT$(zg$,2):re$=RIGHT$(zg$,2)  [7FDA]
1750 PRINT li$;"-";re$:zug$=UPPER$(zg$):
     GOSUB 3750:PRINT CHR$(7)            [240A]
1760 zg$="":NEXT k:s1=s1+1:zk$=""        [A0AE]
1770 :                                   [BE52]
1780 GOTO 1640 '*** Naechster Zug ***    [0E14]
1790 :                                   [CC56]
1800 CLS:s1=1:LOCATE 6,8:PRINT"Weiter ?"
     :LOCATE 7,10:PRINT"(J/N)"           [452C]
1810 w$=INKEY$:IF w$="" THEN 1810        [F848]
1820 IF UPPER$(w$)<>"N" THEN CLS:CLS#3:G
     OSUB 3260:GOTO 1580                 [9916]
1830 n=i:RETURN                          [BF34]
1840 :                                   [CF4E]
1850 REM *** Zufallszahlen erzeugen ****
     ******                              [0278]
1860 :                                   [DD52]
1870 CLS:LOCATE 12,11:PRINT"Bitte Warten
     .":LOCATE 12,13:PRINT"ICH ARBEITE !
     "                                   [22B0]
1880 FOR i=1 TO n                        [F540]
1890 r[i]=INT(n*RND(1)+1)                [C7B0]
1900 IF i=1 THEN 1940                    [B05A]
1910 FOR k=1 TO i-1                      [90EA]
1920 IF r[i]=r[k] THEN 1890              [F782]
1930 NEXT k                              [E06E]
1940 NEXT i                              [0A6C]
1950 :                                   [DC52]
1960 FOR i=1 TO n:richtig[r(i)]=0:NEXT i
                                         [D62C]
1970 GOSUB 2720:CLS:wert=0:punkte=0      [573E]
1980 :                                   [CD58]
1990 REM --------------------------------
     -                                   [642E]
2000 REM *** GANZE VARIANTEN ABFRAGEN **
     *                                   [1B6A]
2010 REM --------------------------------
     -                                   [7B0E]
2020 :                                   [C23C]
2030 LOCATE#2,1,2:PRINT#2,CHR$(150);STRI
     NG$(17,CHR$(154));CHR$(156)         [591A]
2040 LOCATE#2,1,3:PRINT#2,CHR$(149);"ABF
     RAGE VARIANTEN";CHR$(149)           [F882]
2050 LOCATE#2,1,4:PRINT#2,CHR$(147);STRI
     NG$(17,CHR$(154));CHR$(153)         [9B28]
2060 :                                   [CE44]
2070 CLS:FOR i=1 TO n '*** Schleifenanfa
     ng Varianten ***                    [3DF6]
2080 :                                   [C848]
2090 IF richtig[r(i)] THEN 2430 '*** Kei
     n Fehler - Naechste Variante ***    [3724]
2100 richtig[r(i)]=-1:var$[r(i)]=UPPER$(
     var$[r(i)]):p=INSTR(var$[r(i)],"XXX
     X")                                 [0D9E]
2110 IF p MOD 8 = 5 OR p=0 THEN abf=0 EL
     SE abf=1                            [74A0]
2120 :                                   [D43E]
2130 links$=LEFT$(var$(r(i)),p-1)        [12D8]
2140 rechts$=RIGHT$(var$[r(i)],LEN(var$[
     r(i)])-(p+3))                       [A61C]
2150 links$=links$+LEFT$(rechts$,4)      [DF0E]
2160 rechts$=RIGHT$(rechts$,LEN(rechts$)
     -4):zug=0:hz=-1                     [6922]
2170 IF NOT folge THEN 2190              [01E4]
2180 v$=links$+rechts$:IF LEFT$(v$,LEN(f
     olge$))<>folge$ THEN 2430           [4CE4]
2190 GOSUB 5310                          [A7AA]
2200 :                                   [C43C]
2210 FOR j=1 TO LEN(rechts$) STEP 4 '***
      Schleifenanfang Zuege ***          [81A6]
2220 :                                   [C240]
2230 hz$=MID$(rechts$,j,4):li$=LEFT$(hz$
     ,2):re$=RIGHT$(hz$,2)               [00B2]
2240 hz=hz+1:IF hz MOD 2 = 0 THEN p1=6:z
     ug=zug+1 ELSE p1=13                 [FF78]
2250 s2=zug MOD 12:IF s2=0 THEN CLS      [49E0]
```

Listing. Schach – nicht nur für Anfänger

```
2260 LOCATE 2,s2+1:PRINT USING "##";zug;
     :PRINT ".";                                [B8BA]
2270 CLS#3:LOCATE p1+1,s2+1:PRINT CHR$(2
     4)+" ? "+CHR$(24):frage$=""                [C366]
2280 FOR rr=1 TO 4 '*** Zug eingeben ***
                                                [77E4]
2290 z$[rr]=INKEY$:IF z$[rr]="" THEN 229
     0                                          [56D0]
2300 frage$=frage$+z$[rr]:LOCATE#3,7,3:P
     RINT#3,CHR$(24)+" ";frage$;" "+CHR$
     (24)                                       [F1B2]
2310 NEXT rr                                    [F152]
2320 wert=wert+1:IF UPPER$(frage$)=hz$ T
     HEN GOSUB 2470 ELSE GOSUB 2480             [E61A]
2330 LOCATE p1,s2+1:PRINT li$;"-";re$:zu
     g$=li$+re$:GOSUB 3710                      [AE0A]
2340 LOCATE#5,2,3:PRINT #5,punkte;"von";
     wert;"Punkten"                             [F6D6]
2350 IF punkte<>0 THEN proz=INT(punkte/(
     wert/100)) ELSE proz=0                     [FB8E]
2360 LOCATE#5,2,4:PRINT#5,proz;" Prozent
     "                                          [57C0]
2370 :                                          [0D4C]
2380 NEXT j '*** Naechster Zug ***              [1F58]
2390 :                                          [CF50]
2400 CLS#3:LOCATE#3,2,2:PRINT#3,"Naechst
     e Variante"                                [FE68]
2410 LOCATE#3,6,3:PRINT #3,"Tastendruck
     !":CALL &BB18:CLS:GOSUB 3260               [54EC]
2420 :                                          [CA44]
2430 NEXT i:GOTO 2520                           [9B1A]
2440 :                                          [BC48]
2450 REM *** ZUEGE BEWERTEN **********          [389A]
2460 :                                          [BE4C]
2470 punkte=punkte+1:PRINT CHR$(7):RETUR
     N '*** RICHTIGER ZUG EINGEGEBEN ***
                                                [DDC2]
2480 SOUND 150,500,50:richtig[r(i)]=rich
     tig[r(i)]*0:RETURN'*** FALSCHER ZUG
     ***                                        [368E]
2490 :                                          [C352]
2500 REM *** WIEDERHOLUNG ? **********          [F872]
2510 :                                          [CB44]
2520 CLS:LOCATE 3,6:PRINT"Wiederholung ?
     ":LOCATE 7,8:PRINT"(J/N)"                  [0432]
2530 wdhlg$=INKEY$:IF wdhlg$="" THEN 253
     0                                          [9BC4]
2540 IF UPPER$(wdhlg$)="N" THEN 1280 ELS
     E 2030                                     [F318]
2550 :                                          [BF4C]
2560 REM ***  VARIANTEN ABSPEICHERN  ***
     *******                                    [9AFE]
2570 :                                          [BD50]
2580 CLS:LOCATE 5,10:PRINT"Bitte REC und
      PLAY druecken !"                          [4722]
2590 OPENOUT "!chess":PRINT #9,n                [B824]
2600 FOR i=1 TO n:PRINT#9,var$[i]:NEXT i
     :CLOSEOUT:RETURN                           [B170]
2610 :                                          [CF46]
2620 REM ***  VARIANTEN EINLESEN  ******
     ****                                       [4F56]
2630 :                                          [BD4A]
2640 CLS:LOCATE 5,10:PRINT"Bitte PLAY-Ta
     ste druecken !"                            [A6B6]
2650 OPENIN "!chess":INPUT#9,n                  [D222]
2660 FOR i=1 TO n:INPUT#9,var$[i]:NEXT i
     :CLOSEIN:RETURN                            [3DC0]
2670 :                                          [D152]
2680 REM -------------------------------
     ----------                                 [ADF8]
2690 REM *** SCHACHBRETT UND FIGUREN ZEI
     CHNEN ***                                  [130E]
2700 REM -------------------------------
     ----------                                 [BCEA]
2710 :                                          [BF48]
2720 BORDER 6:WINDOW#0,1,19,5,17:PAPER#0
     ,0:PEN#0,1:CLS#0                           [59D2]
2730 WINDOW#1,20,40,1,21:PAPER#1,3:CLS#1
                                                [91EE]
2740 WINDOW#2,1,19,1,4:PAPER#2,0:PEN#2,1
     :CLS#2                                     [DEDA]
2750 WINDOW#3,1,19,22,25:PAPER#3,2:PEN#3
     ,1:CLS#3                                   [32B4]
2760 WINDOW#4,20,40,22,25:PAPER#4,3:PEN#
     4,0:CLS#4                                  [4114]
2770 WINDOW#5,1,19,18,21:PAPER#5,2:PEN#5
     ,0:CLS#5                                   [98C8]
2780 :                                          [DA56]
2790 LOCATE#4,3,23:PRINT#4,"** CHESS-TUT
     OR **"                                     [A350]
2800 LOCATE#4,2,24:PRINT#4,CHR$(164);"19
     85 by P.Meyerbeck"                         [C54C]
2810 :                                          [E34A]
2820 PEN#1,2:PRINT#1,CHR$(22)+CHR$(1)           [A758]
2830 FOR xx=5 TO 17 STEP 4                      [76A2]
2840 FOR yy=5 TO 17 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(143):NEXT yy                [9920]
2850 FOR yy=6 TO 18 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(131):NEXT yy                [7920]
2860 FOR yy=4 TO 16 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(140):NEXT yy,xx             [9752]
2870 FOR xx=7 TO 19 STEP 4                      [7AB2]
2880 FOR yy=3 TO 15 STEP 4::LOCATE#1,xx,
     yy:PRINT#1,CHR$(143):NEXT yy               [AE94]
2890 FOR yy=2 TO 14 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(140):NEXT yy                [7818]
2900 FOR yy=4 TO 16 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(131):NEXT yy,xx             [AC48]
2910 FOR xx=4 TO 16 STEP 4                      [DC9C]
2920 FOR yy=5 TO 17 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(138):NEXT yy                [5B26]
2930 FOR yy=6 TO 18 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(130):NEXT yy                [C51C]
2940 FOR yy=4 TO 16 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(136):NEXT yy,xx             [AA5A]
2950 FOR xx=6 TO 18 STEP 4                      [C8AC]
2960 FOR yy=5 TO 17 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(133):NEXT yy                [1624]
2970 FOR yy=6 TO 18 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(129):NEXT yy                [4A34]
2980 FOR yy=4 TO 16 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(132):LOCATE#1,xx,yy:
     PRINT#1,CHR$(130):NEXT yy                  [8A7C]
2990 FOR yy=2 TO 14 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(136):NEXT yy                [C324]
3000 FOR yy=3 TO 15 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(138):NEXT yy,xx             [2F42]
3010 FOR xx=8 TO 20 STEP 4                      [1B8A]
3020 FOR yy=3 TO 15 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(133):NEXT yy                [4304]
3030 FOR yy=4 TO 16 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(129):NEXT yy                [B314]
3040 FOR yy=2 TO 14 STEP 4:LOCATE#1,xx,y
     y:PRINT#1,CHR$(132):NEXT yy,xx             [F93A]
3050 :                                          [CA44]
3060 FOR ii=120 TO 376 STEP 32:ORIGIN 36
     0,ii:DRAW 256,0,0:NEXT ii                  [48EE]
3070 FOR ii=360 TO 616 STEP 32:ORIGIN ii
     ,376:DRAW 0,-256,0:NEXT ii                 [3B5E]
3080 PEN#1,0:FOR xx=4 TO 20                     [3DCA]
3090 LOCATE#1,xx,2:PRINT#1,CHR$(208):LOC
     ATE#1,xx,18:PRINT#1,CHR$(210)              [AE90]
3100 NEXT xx                                    [4B66]
3110 :                                          [CE3E]
3120 FOR xx=4 TO 20                             [A626]
3130 LOCATE#1,xx,2:PRINT#1,CHR$(208):LOC
     ATE#1,xx,18:PRINT#1,CHR$(210):NEXT
     xx                                         [5D98]
3140 FOR yy=2 TO 18                             [9C38]
3150 LOCATE#1,4,yy:PRINT#1,CHR$(211):LOC
     ATE#1,20,yy:PRINT#1,CHR$(209):NEXT
     yy                                         [4FA2]
3160 :                                          [BD48]
3170 PEN#1,1:LOCATE#1,2,3:PRINT#1,"8":LO
     CATE#1,2,5:PRINT#1,"7"                     [6318]
3180 LOCATE#1,2,7:PRINT#1,"6":LOCATE#1,2
     ,9:PRINT#1,"5"                             [4A86]
3190 LOCATE#1,2,11:PRINT#1,"4":LOCATE#1,
     2,13:PRINT#1,"3"                           [4D2C]
3200 LOCATE#1,2,15:PRINT#1,"2":LOCATE#1,
     2,17:PRINT#1,"1"                           [0024]
3210 LOCATE#1,5,20:PRINT #1,"A":LOCATE#1
     ,7,20:PRINT#1,"B"                          [B1A2]
3220 LOCATE#1,9,20:PRINT#1,"C":LOCATE#1,
     11,20:PRINT#1,"D"                          [56CA]
3230 LOCATE#1,13,20:PRINT#1,"E":LOCATE#1
     ,15,20:PRINT#1,"F"                         [F732]
3240 LOCATE#1,17,20:PRINT#1,"G":LOCATE#1
     ,19,20:PRINT#1,"H"                         [BC4C]
3250 :                                          [BE48]
3260 REM *** Grundstellung definieren **
     ********                                   [D3F0]
3270 :                                          [BC4C]
3280 tw$=CHR$(15)+CHR$(1)+CHR$(240):sw$=
     CHR$(15)+CHR$(1)+CHR$(241)                 [1962]
3290 bw$=CHR$(15)+CHR$(1)+CHR$(242):dw$=
     CHR$(15)+CHR$(1)+CHR$(243)                 [F12A]
3300 kw$=CHR$(15)+CHR$(1)+CHR$(244):lw$=
     CHR$(15)+CHR$(1)+CHR$(245)                 [D644]
3310 ts$=CHR$(15)+CHR$(0)+CHR$(240):ss$=
     CHR$(15)+CHR$(0)+CHR$(241)                 [1342]
3320 bs$=CHR$(15)+CHR$(0)+CHR$(242):ds$=
     CHR$(15)+CHR$(0)+CHR$(243)                 [6F0A]
3330 ks$=CHR$(15)+CHR$(0)+CHR$(244):ls$=
     CHR$(15)+CHR$(0)+CHR$(245)                 [E636]
3340 :                                          [BB48]
3350 a$[1]=tw$:b$[1]=sw$:c$[1]=lw$:d$[1]
     =dw$ '*** 1. und 2. Reihe ***              [3888]
3360 e$[1]=kw$:f$[1]=lw$:g$[1]=sw$:h$[1]
     =tw$                                       [750C]
3370 a$[2]=bw$:b$[2]=bw$:c$[2]=bw$:d$[2]
     =bw$                                       [898A]
```




```
3380 e$[2]=bw$:f$[2]=bw$:g$[2]=bw$:h$[2]
     =bw$                                  [1EAC]
3390 a$[8]=ts$:b$[8]=ss$:c$[8]=ls$:d$[8]
     =ds$ '*** 7. und 8. Reihe ***         [9AC0]
3400 e$[8]=ks$:f$[8]=ls$:g$[8]=ss$:h$[8]
     =ts$                                  [931A]
3410 a$[7]=bs$:b$[7]=bs$:c$[7]=bs$:d$[7]
     =bs$                                  [CD88]
3420 e$[7]=bs$:f$[7]=bs$:g$[7]=bs$:h$[7]
     =bs$                                  [D4AA]
3430 :                                     [D848]
3440 FOR ii=3 TO 6
          '*** 3. bis 6. Reihe leer ***    [2350]
3450 a$[ii]="":b$[ii]="":c$[ii]="":d$[ii
     ]=""                                  [34C0]
3460 e$[ii]="":f$[ii]="":g$[ii]="":h$[ii
     ]=""                                  [ABE2]
3470 NEXT ii                               [893E]
3480 FOR xx=5 TO 19 STEP 2:FOR yy=7 TO 1
     3 STEP 2                              [B0DE]
3490 LOCATE#1,xx,yy:PRINT #1," ":NEXT yy
     ,xx                                   [BC50]
3500 :                                     [C344]
3510 REM *** Figuren aufstellen ********
     **                                    [8DE4]
3520 :                                     [C148]
3530 FOR xx=5 TO 17 STEP 4:FOR yy=5 TO 1
     7 STEP 4:LOCATE#1,xx,yy:PRINT#1,l2$
     :NEXT yy,xx                           [253E]
3540 FOR xx=7 TO 19 STEP 4:FOR yy=3 TO 1
     5 STEP 4:LOCATE#1,xx,yy:PRINT#1,l2$
     :NEXT yy,xx                           [FE40]
3550 FOR xx=7 TO 19 STEP 4:FOR yy=5 TO 1
     7 STEP 4:LOCATE#1,xx,yy:PRINT#1,l1$
     :NEXT yy,xx                           [B748]
3560 FOR xx=5 TO 17 STEP 4:FOR yy=3 TO 1
     5 STEP 4:LOCATE#1,xx,yy:PRINT#1,l1$
     :NEXT yy,xx                           [F83A]
3570 LOCATE#1,5,3:PRINT#1,ts$:LOCATE#1,7
     ,3:PRINT#1,ss$                        [78CC]
3580 LOCATE#1,9,3:PRINT#1,ls$:LOCATE#1,1
     1,3:PRINT#1,ds$                       [E4FE]
3590 LOCATE#1,13,3:PRINT#1,ks$:LOCATE#1,
     15,3:PRINT#1,ls$                      [506C]
3600 LOCATE#1,17,3:PRINT#1,ss$:LOCATE#1,
     19,3:PRINT#1,ts$                      [478C]
3610 FOR ii=5 TO 19 STEP 2:LOCATE#1,ii,5
     :PRINT#1,bs$:NEXT ii                  [3500]
3620 LOCATE#1,5,17:PRINT#1,tw$:LOCATE#1,
     7,17:PRINT#1,sw$                      [87A8]
3630 LOCATE#1,9,17:PRINT#1,lw$:LOCATE#1,
     11,17:PRINT #1,dw$                    [751A]
3640 LOCATE#1,13,17:PRINT #1,kw$:LOCATE#
     1,15,17:PRINT#1,lw$                   [9288]
3650 LOCATE#1,17,17:PRINT#1,sw$:LOCATE#1
     ,19,17:PRINT#1,tw$                    [557A]
3660 FOR ii=5 TO 19 STEP 2:LOCATE#1,ii,1
     5:PRINT#1,bw$:NEXT ii                 [6074]
3670 :                                     [CC54]
3680 PRINT#1,CHR$(22)+CHR$(0):RETURN       [D6F6]
3690 :                                     [0A58]
3700 REM -------------------------         [454C]
3710 REM *** ZUEGE AUSFUEHREN ***          [0882]
3720 REM -------------------------         [3150]
3730 :                                     [C44E]
3740 IF zug$="" THEN 5230                  [EBA2]
3750 PRINT#1,CHR$(22)+CHR$(1)              [2BC0]
3760 x1$=MID$(zug$,1,1):y1$=MID$(zug$,2,
     1):x2$=MID$(zug$,3,1)                 [5120]
3770 y2$=MID$(zug$,4,1):vy=VAL(y1$):vz=V
     AL(y2$)                               [9EAA]
3780 vr$=x1$:GOSUB 3940:x1=w:vr=VAL(y1$)
     :GOSUB 3850:y1=u                      [8AD4]
3790 vr$=x2$:GOSUB 3940:x2=w:vr=VAL(y2$)
     :GOSUB 3850:y2=u                      [5ADE]
3800 vr=VAL(y2$):GOSUB 3850:y2=u           [13A8]
3810 :                                     [CE4C]
3820 l1$=CHR$(15)+CHR$(3)+CHR$(143):l2$=
     CHR$(15)+CHR$(2)+CHR$(143)            [933A]
3830 GOSUB 4030:RETURN                     [D8DE]
3840 :                                     [CD52]
3850 IF vr=1 THEN u=17:RETURN              [F730]
3860 IF vr=2 THEN u=15:RETURN              [6A30]
3870 IF vr=3 THEN u=13:RETURN              [8530]
3880 IF vr=4 THEN u=11:RETURN              [F430]
3890 IF vr=5 THEN u=9:RETURN               [3AE2]
3900 IF vr=6 THEN u=7:RETURN               [A3D0]
3910 IF vr=7 THEN u=5:RETURN               [8AD0]
3920 IF vr=8 THEN u=3:RETURN               [85D0]
3930 :                                     [CC52]
3940 IF vr$="A" THEN w=5:RETURN            [C9BE]
3950 IF vr$="B" THEN w=7:RETURN            [54C6]
3960 IF vr$="C" THEN w=9:RETURN            [EBCE]
3970 IF vr$="D" THEN w=11:RETURN           [F224]
3980 IF vr$="E" THEN w=13:RETURN           [B12C]
3990 IF vr$="F" THEN w=15:RETURN           [0434]
4000 IF vr$="G" THEN w=17:RETURN           [B818]
4010 IF vr$="H" THEN w=19:RETURN           [FF20]
4020 :                                     [C440]
4030 REM *** En-Passent Schlagen *******
     ***                                   [57A0]
4040 :                                     [D244]
4050 ep=INSTR(v$,"B7B5A5B6")               [9488]
4060 IF zug$="A5B6" AND ep<>0 THEN a$[5]
     ="":b$[5]="":b$[6]=bw$:GOTO 4640      [A6DA]
4070 ep=INSTR(v$,"A7A5B5A6")               [C688]
4080 IF zug$="B5A6" AND ep<>0 THEN a$[5]
     ="":b$[5]="":a$[6]=bw$:GOTO 4640      [04DC]
4090 ep=INSTR(v$,"C7C5B5C6")               [EA98]
4100 IF zug$="B5C6" AND ep<>0 THEN b$[5]
     ="":c$[5]="":c$[6]=bw$:GOTO 4640      [3FDA]
4110 ep=INSTR(v$,"B7B5C5B6")               [3A86]
4120 IF zug$="C5B6" AND ep<>0 THEN b$[5]
     ="":c$[5]="":b$[6]=bw$:GOTO 4640      [15DC]
4130 ep=INSTR(v$,"D7D5C5D6")               [8E96]
4140 IF zug$="C5D6" AND ep<>0 THEN c$[5]
     ="":d$[5]="":d$[6]=bw$:GOTO 4640      [32EC]
4150 ep=INSTR(v$,"C7C5D5C6")               [6496]
4160 IF zug$="D5C6" AND ep<>0 THEN c$[5]
     ="":d$[5]="":c$[6]=bw$:GOTO 4640      [E0EE]
4170 ep=INSTR(v$,"E7E5D5E6")               [78A6]
4180 IF zug$="D5E6" AND ep<>0 THEN d$[5]
     ="":e$[5]="":e$[6]=bw$:GOTO 4640      [89FE]
4190 ep=INSTR(v$,"D7D5E5D6")               [9EA6]
4200 IF zug$="E5D6" AND ep<>0 THEN d$[5]
     ="":e$[5]="":d$[6]=bw$:GOTO 4640      [A3EE]
4210 ep=INSTR(v$,"F7F5E5F6")               [6CA4]
4220 IF zug$="E5F6" AND ep<>0 THEN e$[5]
     ="":f$[5]="":f$[6]=bw$:GOTO 4640      [76FE]
4230 ep=INSTR(v$,"E7E5F5E6")               [42A4]
4240 IF zug$="F5E6" AND ep<>0 THEN e$[5]
     ="":f$[5]="":e$[6]=bw$:GOTO 4640      [8900]
4250 ep=INSTR(v$,"G7G5F5G6")               [36B4]
4260 IF zug$="F5G6" AND ep<>0 THEN f$[5]
     ="":g$[5]="":g$[6]=bw$:GOTO 4640      [F010]
4270 ep=INSTR(v$,"F7F5G5F6")               [64B4]
4280 IF zug$="G5F6" AND ep<>0 THEN f$[5]
     ="":g$[5]="":f$[6]=bw$:GOTO 4640      [B612]
4290 ep=INSTR(v$,"H7H5G5H6")               [5CC4]
4300 IF zug$="G5H6" AND ep<>0 THEN g$[5]
     ="":h$[5]="":h$[6]=bw$:GOTO 4640      [0B10]
4310 ep=INSTR(v$,"G7G5H5G6")               [64B2]
4320 IF zug$="H5G6" AND ep<>0 THEN g$[5]
     ="":h$[5]="":g$[6]=bw$:GOTO 4640      [AD12]
4330 ep=INSTR(v$,"G2G4H4G3")               [82A2]
4340 IF zug$="H4G3" AND ep<>0 THEN h$[4]
     ="":g$[4]="":g$[3]=bs$:GOTO 4640      [57FC]
4350 ep=INSTR(v$,"H2H4G4H3")               [8CAA]
4360 IF zug$="G4H3" AND ep<>0 THEN h$[4]
     ="":g$[4]="":h$[3]=bs$:GOTO 4640      [1202]
4370 ep=INSTR(v$,"F2F4G4F3")               [10A2]
4380 IF zug$="G4F3" AND ep<>0 THEN g$[4]
     ="":f$[4]="":f$[3]=bs$:GOTO 4640      [94FA]
4390 ep=INSTR(v$,"G2G4F4G3")               [A2AA]
4400 IF zug$="F4G3" AND ep<>0 THEN g$[4]
     ="":f$[4]="":g$[3]=bs$:GOTO 4640      [4CEE]
4410 ep=INSTR(v$,"E2E4F4E3")               [6490]
4420 IF zug$="F4E3" AND ep<>0 THEN f$[4]
     ="":e$[4]="":e$[3]=bs$:GOTO 4640      [3DE6]
4430 ep=INSTR(v$,"F2F4E4F3")               [2E98]
4440 IF zug$="E4F3" AND ep<>0 THEN f$[4]
     ="":e$[4]="":f$[3]=bs$:GOTO 4640      [6BEC]
4450 ep=INSTR(v$,"D2D4E4D3")               [5A90]
4460 IF zug$="E4D3" AND ep<>0 THEN e$[4]
     ="":d$[4]="":d$[3]=bs$:GOTO 4640      [F0E4]
4470 ep=INSTR(v$,"E2E4D4E3")               [5498]
4480 IF zug$="D4E3" AND ep<>0 THEN e$[4]
     ="":d$[4]="":e$[3]=bs$:GOTO 4640      [E6EA]
4490 ep=INSTR(v$,"C2C4D4C3")               [8C90]
4500 IF zug$="D4C3" AND ep<>0 THEN d$[4]
     ="":c$[4]="":c$[3]=bs$:GOTO 4640      [DBD0]
4510 ep=INSTR(v$,"D2D4C4D3")               [8C86]
4520 IF zug$="C4D3" AND ep<>0 THEN d$[4]
     ="":c$[4]="":d$[3]=bs$:GOTO 4640      [C9D6]
4530 ep=INSTR(v$,"B2B4C4B3")               [587E]
4540 IF zug$="C4B3" AND ep<>0 THEN c$[4]
     ="":b$[4]="":b$[3]=bs$:GOTO 4640      [44CE]
4550 ep=INSTR(v$,"C2C4B4C3")               [5686]
4560 IF zug$="B4C3" AND ep<>0 THEN c$[4]
     ="":b$[4]="":c$[3]=bs$:GOTO 4640      [12D4]
4570 ep=INSTR(v$,"A2A4B4A3")               [9E7E]
4580 IF zug$="B4A3" AND ep<>0 THEN b$[4]
     ="":a$[4]="":a$[3]=bs$:GOTO 4640      [C1CC]
4590 ep=INSTR(v$,"B2B4A4B3")               [5486]
4600 IF zug$="A4B3" AND ep<>0 THEN b$[4]
     ="":a$[4]="":b$[3]=bs$:GOTO 4640      [63C0]
4610 :                                     [BD4A]
4620 GOTO 4710 '*** Kein En-Passen-Zug g
     efunden ***                           [C3A8]
4630 :                                     [CF4E]
4640 IF vy=5 THEN bauer$=bw$ ELSE bauer$
     =bs$                                  [FA58]
```

Listing. Schach – nicht nur für Anfänger (Fortsetzung)

```
4650 IF ASC(x1$)<ASC(x2$) THEN sp=x1+2 E
     LSE sp=x1-2                           [9FD6]
4660 xx$=x1$:vv=vy:GOSUB 5250:LOCATE#1,x
     1,y1:PRINT#1,leer$                    [C614]
4670 IF leer$=l1$ THEN leer$=l2$:GOTO 46
     90                                    [EC14]
4680 IF leer$=l2$ THEN leer$=l1$           [5D4A]
4690 LOCATE#1,sp,y1:PRINT#1,leer$:LOCATE
     #1,x2,y2:PRINT#1,bauer$:GOTO 5230    [CC82]
4700 :                                     [D04A]
4710 REM *** Rochade **********             [C510]
4720 :                                     [BE4E]
4730 IF zug$="E1G1" AND e$[vy]=kw$ AND f
     $[vy]="" AND g$[vy]="" THEN LOCATE#
     1,13,17:PRINT#1,l2$:LOCATE#1,19,17:
     PRINT#1,l1$:LOCATE #1,15,17:PRINT#1
     ,tw$:LOCATE#1,17,17:PRINT#1,kw$:h$[
     vz]="":g$[vz]=kw$:f$(vz)=tw$:e$[vz]
     ="":GOTO 5230                         [A41A]
4740 IF zug$="E1C1" AND e$[vy]=kw$ AND b
     $[vy]="" AND c$[vy]="" AND d$[vy]="
     " THEN LOCATE#1,13,17:PRINT#1,l2$:L
     OCATE#1,5,17:PRINT#1,l2$:LOCATE#1,1
     1,17:PRINT#1,tw$:LOCATE#1,9,17:PRIN
     T#1,kw$:e$[vz]="":a$[vz]="":b$[vz]=
     "":c$[vz]=kw$:d$[vz]=tw$:GOTO 5230    [0102]
4750 IF zug$="E8G8" AND e$[vy]=ks$ AND f
     $[vy]="" AND g$[vy]="" THEN LOCATE#
     1,13,3:PRINT#1,l1$:LOCATE#1,19,3:PR
     INT#1,l2$:LOCATE#1,15,3:PRINT#1,ts$
     :LOCATE#1,17,3:PRINT#1,ks$:h$[vz]="
     ":g$[vz]=ks$:f$[vz]=ts$:e$[vz]="":G
     OTO 5230                              [CEF8]
4760 IF zug$="E8C8" AND e$[vy]=ks$ AND b
     $[vy]="" AND c$[vy]="" AND d$[vy]="
     " THEN 4770 ELSE 4810                 [18A4]
4770 LOCATE#1,13,3:PRINT#1,l1$:LOCATE#1,
     5,3:PRINT #1,l1$:LOCATE #1,9,3:PRIN
     T#1,ks$:LOCATE#1,11,3:PRINT#1,ts$:P
     EN 1:e$[vz]="":a$[vz]="":b$[vz]="":
     c$[vz]=ks$:d$[vz]=ts$:GOTO 5230       [9FFA]
4780 :                                     [D85A]
4790 REM *** BAUERNUMWANDLUNG ***          [6FE6]
4800 :                                     [164C]
4810 IF zug$="A2A1" AND a$[2]=bs$ THEN L
     OCATE#1,5,15::PRINT#1,l1$:LOCATE#1,
     5,17:PRINT#1,l2$:LOCATE#1,5,17:PRIN
     T#1,ds$:a$[2]="":a$[1]=ds$:GOTO 523
     0                                     [B818]
4820 IF zug$="B2B1" AND b$[2]=bs$ THEN L
     OCATE#1,7,15:PRINT#1,l2$:LOCATE#1,7
     ,17:PRINT#1,l1$:LOCATE#1,7,17:PRINT
     #1,ds$:b$[2]="":b$[1]=ds$:GOTO 5230
                                           [29BC]
4830 IF zug$="C2C1" AND c$[2]=bs$ THEN L
     OCATE#1,9,15:PRINT#1,l1$:LOCATE#1,9
     ,17:PRINT#1,l2$:LOCATE#1,9,17:PRINT
     #1,ds$:c$[2]="":c$[1]=ds$:GOTO 5230
                                           [2ED4]
4840 IF zug$="D2D1" AND d$[2]=bs$ THEN L
     OCATE#1,11,15:PRINT#1,l2$:LOCATE#1,
     11,17:PRINT#1,l1$:LOCATE#1,11,17:PR
     INT#1,ds$:d$[2]="":d$[1]=ds$:GOTO 5
     230                                   [72D6]
4850 IF zug$="E2E1" AND e$[2]=bs$ THEN L
     OCATE#1,13,15:PRINT#1,l1$:LOCATE#1,
     13,17:PRINT#1,l2$:LOCATE#1,13,17:PR
     INT#1,ds$:e$[2]="":e$[1]=ds$:GOTO 5
     230                                   [C8EE]
4860 IF zug$="F2F1" AND f$[2]=bs$ THEN L
     OCATE#1,15,15:PRINT#1,l2$:LOCATE#1,
     15,17:PRINT#1,l1$:LOCATE#1,15,17:PR
     INT#1,ds$:f$[2]="":f$[1]=ds$:GOTO 5
     230                                   [4106]
4870 IF zug$="G2G1" AND g$[2]=bs$ THEN L
     OCATE#1,17,15:PRINT#1,l1$:LOCATE#1,
     17,17:PRINT#1,l2$:LOCATE#1,17,17:PR
     INT#1,ds$:g$[2]="":g$[1]=ds$:GOTO 5
     230                                   [A71E]
4880 IF zug$="H2H1" AND h$[2]=bs$ THEN L
     OCATE#1,19,15:PRINT#1,l2$:LOCATE#1,
     19,17:PRINT#1,l1$:LOCATE#1,19,17:PR
     INT#1,ds$:h$[2]="":h$[1]=ds$:GOTO 5
     230                                   [7F36]
4890 IF zug$="A7A8" AND a$[7]=bw$ THEN L
     OCATE#1,5,5:PRINT#1,l2$:LOCATE#1,5,
     37:PRINT#1,l1$:LOCATE#1,5,3:PRINT#1
     ,dw$:a$[7]="":a$[8]=dw$:GOTO 5230     [573E]
4900 IF zug$="B7B8" AND b$[7]=bw$ THEN L
     OCATE#1,7,5:PRINT#1,l1$:LOCATE#1,7,
     3:PRINT#1,l2$:LOCATE#1,7,3:PRINT#1,
     dw$:b$[7]="":b$[8]=dw$:GOTO 5230      [11D6]
4910 IF zug$="C7C8" AND c$[7]=bw$ THEN L
     OCATE#1,9,5:PRINT#1,l2$:LOCATE#1,9,
     3:PRINT#1,l1$:LOCATE#1,9,3:PRINT#1,
     dw$:b$[7]="":c$[8]=dw$:GOTO 5230      [08EC]
4920 IF zug$="D7D8" AND d$[7]=bw$ THEN L
     OCATE#1,11,5:PRINT#1,l1$:LOCATE#1,1
     1,3:PRINT#1,l2$:LOCATE#1,11,3:PRINT
     #1,dw$:d$[7]="":d$[8]=dw$:GOTO 5230
                                           [2FF0]
4930 IF zug$="E7E8" AND e$[7]=bw$ THEN L
     OCATE#1,13,5:PRINT#1,l2$:LOCATE#1,1
     3,3:PRINT#1,l1$:LOCATE#1,13,3:PRINT
     #1,dw$:e$[7]="":e$[8]=dw$:GOTO 5230
                                           [E308]
4940 IF zug$="F7F8" AND f$[7]=bw$ THEN L
     OCATE#1,15,5:PRINT#1,l1$:LOCATE#1,1
     5,3:PRINT#1,l2$:LOCATE#1,15,3:PRINT
     #1,dw$:f$[7]="":f$[8]=dw$:GOTO 5230
                                           [BA20]
4950 IF zug$="G7G8" AND g$[7]=bw$ THEN L
     OCATE#1,17,5:PRINT#1,l2$:LOCATE#1,1
     7,3:PRINT#1,l1$:LOCATE#1,17,3:PRINT
     #1,dw$:g$[7]="":g$[8]=dw$:GOTO 5230
                                           [7138]
4960 IF zug$="H7H8" AND h$[7]=bw$ THEN L
     OCATE#1,19,5:PRINT#1,l1$:LOCATE#1,1
     9,3:PRINT#1,l2$:LOCATE#1,19,3:PRINT
     #1,dw$:h$[7]="":h$[8]=dw$:GOTO 5230
                                           [CC50]
4970 :                                     [BF5C]
4980 REM *** Normale Zuege ***             [FD46]
4990 :                                     [D160]
5000 IF x1$="A" THEN feld$=a$[vy]:a$[vy]
     =""                                   [1B42]
5010 IF x1$="B" THEN feld$=b$[vy]:b$[vy]
     =""                                   [B94A]
5020 IF x1$="C" THEN feld$=c$[vy]:c$[vy]
     =""                                   [9B52]
5030 IF x1$="D" THEN feld$=d$[vy]:d$[vy]
     =""                                   [B15A]
5040 IF x1$="E" THEN feld$=e$[vy]:e$[vy]
     =""                                   [1362]
5050 IF x1$="F" THEN feld$=f$[vy]:f$[vy]
     =""                                   [D16A]
5060 IF x1$="G" THEN feld$=g$[vy]:g$[vy]
     =""                                   [9372]
5070 IF x1$="H" THEN feld$=h$[vy]:h$[vy]
     =""                                   [997A]
5080 :                                     [CF4E]
5090 xx$=x1$:vv=vy:GOSUB 5250:LOCATE#1,x
     1,y1:PRINT#1,leer$                    [8210]
5100 xx$=x2$:vv=vz:GOSUB 5250:LOCATE#1,x
     2,y2:PRINT#1,leer$:LOCATE#1,x2,y2:P
     RINT#1,feld$                          [FBFA]
5110 :                                     [D442]
5120 FOR ii=1 TO 8                         [0294]
5130 IF x2$="A" AND vz=ii THEN a$[vz]=fe
     ld$                                   [52A4]
5140 IF x2$="B" AND vz=ii THEN b$[vz]=fe
     ld$                                   [A6AA]
5150 IF x2$="C" AND vz=ii THEN c$[vz]=fe
     ld$                                   [F6B0]
5160 IF x2$="D" AND vz=ii THEN d$[vz]=fe
     ld$                                   [2AB6]
5170 IF x2$="E" AND vz=ii THEN e$[vz]=fe
     ld$                                   [52BC]
5180 IF x2$="F" AND vz=ii THEN f$[vz]=fe
     ld$                                   [8EC2]
5190 IF x2$="G" AND vz=ii THEN g$[vz]=fe
     ld$                                   [B6C8]
5200 IF x2$="H" AND vz=ii THEN h$[vz]=fe
     ld$                                   [9FBC]
5210 NEXT ii                               [B132]
5220 :                                     [DD46]
5230 RETURN                                [C194]
5240 :                                     [DF4A]
5250 IF (xx$="A" OR xx$="C" OR xx$="E" O
     R xx$="G") AND vv MOD 2 = 0 THEN le
     er$=l1$                               [E554]
5260 IF (xx$="A" OR xx$="C" OR xx$="E" O
     R xx$="G") AND vv MOD 2 <> 0 THEN l
     eer$=l2$                              [EFD2]
5270 IF (xx$="B" OR xx$="D" OR xx$="F" O
     R xx$="H") AND vv MOD 2 = 0 THEN le
     er$=l2$                               [7462]
5280 IF (xx$="B" OR xx$="D" OR xx$="F" O
     R xx$="H") AND vv MOD 2 <> 0  THEN
     leer$=l1$                             [5D1C]
5290 RETURN                                [95A0]
5300 :                                     [C544]
5310 FOR j=1 TO LEN(links$) STEP 4         [B4A0]
5320 hz$=MID$(links$,j,4)                  [F46E]
5330 hz=hz+1:IF hz MOD 2 = 0 THEN p1=6:z
     ug=zug+1 ELSE p1=13                   [FE7E]
5340 s2= zug MOD 12:IF s2=0 THEN CLS       [6626]
5350 LOCATE 2,s2+1:PRINT USING "##";zug;
     :PRINT ".".                           [76C0]
5360 li$=LEFT$(hz$,2):re$=RIGHT$(hz$,2)    [8BE2]
5370 LOCATE p1,s2+1:PRINT li$;"-";re$      [83AE]
5380 zug$=li$+re$:GOSUB 3710:FOR warte=1
      TO 1000:NEXT                         [31B4]
5390 NEXT j:RETURN                         [1EA8]
```

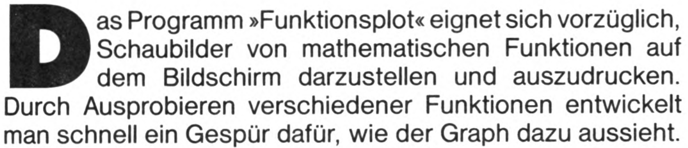
Listing. Schach – nicht nur für Anfänger (Schluß)



Mathematik anschaulich



Für den Laien undurchschaubare mathematische Formeln stellt dieses Programm klar und zum »Anfassen« dar. Dabei ist es so flexibel, wie man es sich nur wünschen kann.

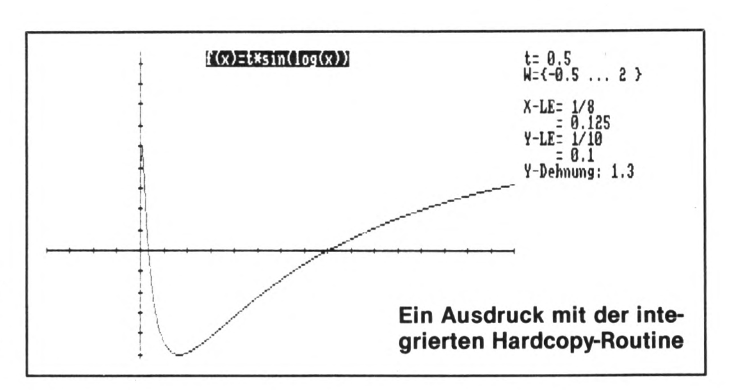


Das Programm »Funktionsplot« eignet sich vorzüglich, Schaubilder von mathematischen Funktionen auf dem Bildschirm darzustellen und auszudrucken. Durch Ausprobieren verschiedener Funktionen entwickelt man schnell ein Gespür dafür, wie der Graph dazu aussieht.

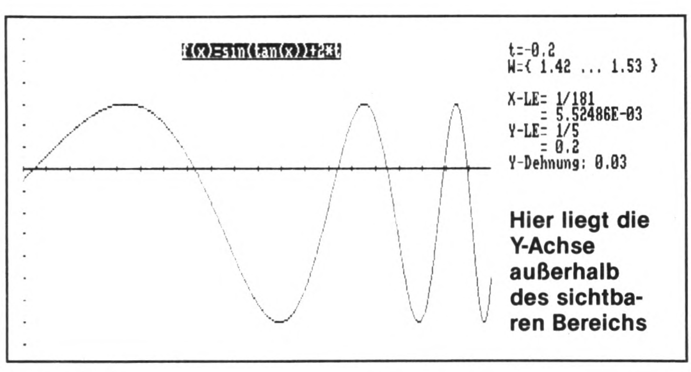
Die Funktionen können in Abhängigkeit von einem Parameter »t« angegeben werden.

Nach Vorgabe der Funktion werden verschiedene Angaben verlangt:

1. Der Wert des Parameters »t« (Bei »ENTER« Standardwert 0).
2. Der Wertebereich, für den die Funktion zutrifft (von, bis).



Ein Ausdruck mit der integrierten Hardcopy-Routine



Hier liegt die Y-Achse außerhalb des sichtbaren Bereichs



```
1   '*****************************         [3060]
2   '*                           *         [2F5A]
3   '*          FUNKTION         *         [7038]
4   '*                           *         [155E]
5   '*          (C) 1985         *         [B776]
6   '*      Timo Breidenstein    *         [574C]
7   '*   Reichenbacher Str. 28   *         [428C]
8   '*      7290 Freudenstadt    *         [3CDC]
9   '*                           *         [6C68]
10  '*        Hardcopy-RSX       *         [2340]
11  '*     von Helmut Tischer    *         [DFA2]
12  '* aus Happy-Computer 12/85  *         [1F88]
13  '*                           *         [007E]
14  '*****************************         [5088]
50 '------ MC-PROGRAMME POKEN ------       [8118]
60 MODE 2:IF PEEK(40000)<>&3E THEN MEMOR
   Y 39999:GOSUB 2000:GOTO 3000            [34B2]
100 '------ MENUE ------                   [925C]
110 IF f$=""THEN GOSUB 200                 [80DA]
120 WINDOW#0,61,80,12,25:CLS               [2886]
130 PRINT CHR$(24);"{5 SPACE}M e n u e{5
     SPACE}";CHR$(24),,"<1> Neue Funktio
    n","<2> Neue Wertemenge","<3> Neue Y
    -Dehnung","<4> Neues t"                [865C]
140 PRINT "<5> Graph zeichnen","<6> Hard
    copy","<7> Funktionswerte{6 SPACE}ab
    fragen","<8> Ende",,"Ihre Wahl (1-8)
    "                                      [3E12]
150 IF INKEY$<>""THEN 150                  [4258]
160 e$=INKEY$:IF e$<"1"OR e$>"8" THEN LO
    CATE 17,13:PRINT"?":PRINT CHR$(24):F
    OR n=1 TO 100:NEXT:GOTO 160:ELSE PEN
     1:PAPER 0:CLS                         [37F2]
170 ON VAL(e$) GOSUB 200,400,500,300,700
    ,1000,600,990                          [1308]
180 GOTO 100                               [B846]
200 '------ EINGABE DER FUNKTION ------    [DF52]
210 MODE 2:INPUT"Funktion :{5 SPACE}f(x)
    =",f$:IF f$="" THEN 210                [1630]
220 KEY 139,"240 DEF FNt(x)="+f$+CHR$(13
    )+"goto 240"+CHR$(13):PEN 0            [3B7E]
230 POKE &B516,0:POKE &B517,64:POKE &B53
    C,20:POKE &B53D,1:POKE &B53E,2:POKE
    &B53F,0:STOP                           [5246]
240 DEF FNt(x)=LOG(x)+SIN(x)*LOG(x)+COS(
    x)                                     [AFD4]
250 POKE &B540,0                           [C6C6]
260 CLS:PEN 1:KEY 139,CHR$(13):PRINT "Ne
    ue Funktion :{5 SPACE}f(x)="f$:PRINT
    :PRINT                                 [E2DC]
270 GOSUB 300:GOSUB 400:GOSUB 500:GOSUB
    700                                    [C574]
280 GOTO 110                               [0A4A]
300 '------ EINGABE VON t ------           [7190]
310 INPUT"t=",t                            [E8B2]
320 RETURN                                 [AC2A]
400 '------ EINGABE DES WERTEBEREICHS --
    ----                                   [4A14]
410 INPUT;"Wertebereich:{7 SPACE}von ",u
    :INPUT" bis ",h:IF h-u<=0 THEN 410     [1616]
420 RETURN                                 [A92C]
500 '------ EINGABE DER Y-DEHNUNG ------
                                           [B89A]
510 INPUT"Dehnung in{10 SPACE}y-Richtung
    : ",d:IF d=0 THEN d=1                  [B0B2]
520 RETURN                                 [AA2E]
600 '------ FUNKTIONSWERTE ABFRAGEN ----
    --                                     [1CAE]
610 ON ERROR GOTO 640                      [2422]
620 INPUT"x=",x$:IF LOWER$(x$)="m" THEN
    RETURN ELSE x=VAL(x$)                  [8F6E]
630 PRINT"f(";STR$(x);")=";FNt(x):PRINT:
    GOTO 620                               [9D38]
640 PRINT CHR$(11);"Nicht definiert !":P
    RINT:RESUME 610                        [EC68]
700 '------ GRAPH ZEICHNEN ------          [A940]
710 MODE 2                                 [F25E]
720 b=h-u:xp=470/b:yp=d*400/b:g=u*xp+470
                                           [1856]
730 ORIGIN -u*xp,200:MOVE u*xp,0:DRAW g,
    0:IF g>=0 THEN MOVE 0,199:DRAW 0,-19
    9                                      [038E]
740 fx=1+INT(b/20):IF b/20<0.5 THEN fx=1
    /INT(20/b)                             [E24A]
750 fy=1+INT(b/(20*ABS(d))):IF b/(20*ABS
    (d))<0.5 THEN fy=1/INT((20*ABS(d))/b
    )                                      [9D74]
760 xe=fx*xp:ye=fy*yp                      [801C]
770 p=30-(5+LEN(f$))/2:IF p<1 THEN p=1     [F0B4]
780 xe$="":IF fx>=1 THEN x$=STR$(fx) ELS
    E x$=" 1/"+MID$(STR$(1/fx),2):xe$="{
    4 SPACE}="+STR$(fx)                    [0A84]
790 ye$="":IF fy>=1 THEN y$=STR$(fy) ELS
    E y$=" 1/"+MID$(STR$(1/fy),2):ye$="{
    4 SPACE}="+STR$(ABS(fy))               [51E4]
800 LOCATE#2,p,1:PRINT#2,CHR$(24);"f(x)=
    ";f$;CHR$(24):WINDOW#1,61,80,1,11:PR
    INT#1,"t="t:PRINT#1,"W={"u"..."h"}":
    PRINT#1                                [DB54]
810 PRINT#1,"X-LE=";x$,xe$:PRINT#1,"Y-LE
    =";y$,ye$:PRINT#1,"Y-Dehnung:"d        [B37C]
820 FOR x=0 TO h*xp STEP xe:MOVE x,2:DRA
    W x,-2:NEXT:FOR x=0 TO u*xp STEP -xe
    :IF x>g THEN 830 ELSE MOVE x,2:DRAW
    x,-2                                   [4B1C]
830 NEXT                                   [5AF4]
840 IF SGN(u)=SGN(h) THEN j=u*xp ELSE j=
    0                                      [B6A8]
850 FOR x=0 TO 199 STEP ABS(ye):MOVE j+2
    ,x:DRAW j-2,x:NEXT:FOR x=0 TO -199 S
    TEP -ABS(ye):MOVE j+2,x:DRAW j-2,x:N
    EXT                                    [9916]
860 x=u*xp                                 [1AF4]
870 ON ERROR GOTO 950                      [C93A]
880 MOVE x,FNt(x/xp)*yp                    [BD6C]
890 ON ERROR GOTO 960                      [9A40]
900 FOR x=x TO g                           [166E]
910 y=FNt(x/xp)*yp                         [7ED6]
920 IF ABS(y)>=199 OR (SGN(y)<>SGN(YPOS)
     AND ABS(YPOS)=199) THEN 970 ELSE DR
    AW  x,y                                [DEE4]
930 NEXT                                   [89F6]
940 LOCATE#2,p,1:PRINT#2,CHR$(24);"f(x)=
```

Listing. Mathematische Funktionen

```
     "f$;CHR$(24):RETURN                     [828A]
950  x=x+1:IF x>g THEN 940 ELSE RESUME 88
     0                                       [F9A6]
960  IF ERR=11 THEN RESUME NEXT ELSE RESU
     ME 970                                  [271E]
970  IF ABS(YPOS)<199 THEN DRAW x,SGN(YPO
     S)*199 ELSE PLOT x,SGN(y)*199           [82BA]
980  GOTO 930                                [BD6C]
990  MODE 2:END                              [7094]
1000 '------ HARDCOPY ------                 [9C7C]
1010 CLS:CALL 40000                          [315C]
1020 IF PEEK(40100)=&FF THEN PRINT CHR$(
     7);CHR$(24);"Drucker nicht bereit";
     CHR$(24):FOR n=1 TO 3000:NEXT:RETUR
     N                                       [D9DA]
1030 |HARDCOPY:RETURN                        [FFA8]
2000 '------ DRUCKER BEREIT? ------          [441E]
2010 FOR n=40000 TO 40016:READ w:POKE n,
     w:NEXT:RETURN                           [2702]
2020 DATA &3e,&00,&32,&a4,&9c,&cd,&2e,&b
     d,&38,&01,&c9,&3e,&ff,&32,&a4,&9c,&
     c9                                      [2B28]
3000 '------ HARDCOPY-RSX ------             [86D4]
3010 OPENOUT"dummy":MEMORY HIMEM-1:CLOSE
     OUT                                     [B74E]
3020 st=HIMEM-335:MEMORY st-1                [9D72]
3030 RESTORE 3110                            [13DE]
3040 FOR i=0 TO 335 STEP 8                   [FFEC]
3050 pruef=0                                 [13EE]
3060 FOR j=i TO i+7:READ d$:d=VAL("&"+d$
     ):POKE st+j,d:pruef=pruef+d:NEXT        [7C60]
3070 READ p:IF pruef<>p THEN PRINT"Fehle
     r in Zeile"3110+i/8*10". Bitte Verb
     essern und dann GOTO 3030 eingeben.
     ":END                                   [B490]
3080 NEXT                                    [F354]
3090 FOR i=1 TO 27:READ d$:d=VAL("&"+d$)
     :z=st+PEEK(st+d)+256*PEEK(st+d+1):P
     OKE st+d+1,INT(z/256):POKE st+d,z-2
     56*INT(z/256):NEXT                      [7A0C]
3100 CALL HIMEM+1:GOTO 100                   [0E20]
3110 DATA 01,09,00,21,17,00,c3,d1, 470       [5574]
3120 DATA bc,0e,00,c3,1b,00,48,41, 561       [B3A0]
3130 DATA 52,44,43,4f,50,d9,00,00, 593       [27A8]
3140 DATA 00,00,00,ed,73,31,01,3a, 460       [D9D0]
3150 DATA c8,b1,87,fe,01,ce,00,87,1108       [935A]
3160 DATA 32,29,01,0e,0d,cd,13,01, 344       [724A]
3170 DATA 0e,0a,cd,13,01,3e,02,cd, 518       [4E6C]
3180 DATA 8a,00,3e,7f,32,28,01,2a, 460       [5E62]
3190 DATA c9,b1,7c,f6,c0,67,06,1d,1078       [026E]
3200 DATA 05,20,05,3e,78,32,28,01, 315       [1436]
3210 DATA 04,c5,cd,9f,00,3a,c8,b1,1000       [5B8A]
3220 DATA fe,02,cc,9f,00,11,00,08, 644       [B7B6]
3230 DATA a7,ed,52,7c,f6,3f,3c,28,1019       [EFC8]
3240 DATA 08,11,b0,3f,19,7c,f6,f8, 907       [E6EC]
3250 DATA 67,0e,0a,cd,13,01,0e,0d, 379       [E31E]
3260 DATA cd,13,01,e5,3e,42,cd,1e, 817       [AF8C]
3270 DATA bb,c2,23,01,e1,c1,10,c0,1043       [4016]
3280 DATA 3e,03,21,33,01,4e,23,06, 269       [42A2]
3290 DATA 00,09,3d,20,f8,46,23,4e, 533       [2016]
3300 DATA 23,cd,13,01,10,f9,c9,e5, 955       [36D4]
3310 DATA 21,33,01,3a,c8,b1,fe,02, 776       [59BC]
3320 DATA 28,05,4e,23,06,00,09,cd, 378       [140E]
3330 DATA 95,00,e1,3a,c8,b1,fe,02,1065       [7F44]
3340 DATA 06,28,28,02,06,50,c5,e5, 600       [68A6]
3350 DATA 11,2a,01,3e,07,ed,a0,01, 527       [C1A6]
3360 DATA ff,07,09,30,0a,01,50,c0, 602       [E44E]
3370 DATA 09,47,7c,e6,c7,67,78,3d, 917       [50BC]
3380 DATA 20,eb,21,cf,b1,3a,29,01, 784       [EC26]
3390 DATA 47,c5,11,2a,01,06,07,1a, 367       [BE02]
3400 DATA 13,a6,fe,01,3f,cb,11,10, 739       [FF18]
3410 DATA f6,3a,28,01,a1,4f,cd,13, 809       [A22A]
3420 DATA 01,3a,c8,b1,a7,cc,13,01, 827       [2914]
3430 DATA c1,23,10,dd,e1,c1,7c,e6,1237       [49A4]
3440 DATA f8,4f,23,7c,e6,07,b1,67,1003       [0B04]
3450 DATA 10,ac,c9,79,cd,2b,bd,d8,1163       [107E]
3460 DATA c5,e5,3e,42,cd,1e,bb,e1,1201       [B664]
3470 DATA c1,28,f0,ed,7b,31,01,c9,1084       [2A58]
3480 DATA 00,00,00,00,00,00,00,00,   0       [DB32]
3490 DATA 00,00,00,04,1b,4c,40,01, 172       [176E]
3500 DATA 04,1b,4b,40,01,05,1b,41, 268       [D7E2]
3510 DATA 07,1b,32,05,1b,41,0c,1b, 220       [9E32]
3520 DATA 32,00,00,00,00,00,00,00,  50       [FB5C]
3530 DATA 0001,0004,0009,000c                [DFD4]
3540 DATA 001d,0029,002e,0033                [2C4E]
3550 DATA 0038,003d,004e,0053                [E35C]
3560 DATA 005b,0074,0079,0082                [5E10]
3570 DATA 008b,009a,00a1,00b0                [D30A]
3580 DATA 00c1,00de,00e3,00f2                [2D76]
3590 DATA 00f7,00fe,0125                     [FAE6]
```

Listing. Mathematische Funktionen (Schluß)

3. Die Dehnung in Y-Richtung: Sie gibt das Verhältnis zwischen einer Längeneinheit (LE) auf der X-Achse und einer LE auf der Y-Achse an. Wird hierfür 1 eingegeben, so sind beide gleich lang. Durch Werte größer 1 wird das Schaubild vertikal gedehnt, durch Werte kleiner 1 gestaucht. Negative Werte führen zu einer Spiegelung an der X-Achse.

Wird anstelle einer Eingabe gleich ENTER gedrückt, so setzt das Programm als Standardwert für die Y-Dehnung eine 1.

Die Koordinatenachsen werden automatisch in sinnvolle Abschnitte eingeteilt. Liegt der Ursprung und damit auch die Y-Achse außerhalb des gewählten Wertebereichs, so wird anstelle einer kompletten Y-Achse nur ihre Einteilung (Querstriche) am Bildrand angedeutet: links, wenn die Achse unterhalb des Wertebereichs liegt, und rechts, wenn sie oberhalb davon liegt.

Eine Anzeige für beide Achsen gibt an, welchem Zahlenwert der Abstand zwischen je zwei Querstrichen auf der Achse entspricht.

Verläuft die Kurve oberhalb des oberen Bildschirmrandes, ist dies dadurch ersichtlich, daß sie als gerade Linie am oberen Bildschirmrand erscheint. Entsprechendes gilt, wenn sie unterhalb des unteren Bildrandes verläuft. Liegen Teile des Wertebereichs außerhalb des Definitionsbereichs der Funktion, so wird nichts gezeichnet (zum Beispiel Logarithmusfunktion für negative Werte). Wenn der Graph gezeichnet ist, erscheint daneben ein Auswahlmenü:

1. Neue Funktion eingeben
2. Neue Wertemenge
3. Neue Y-Dehnung
4. Neuer Wert für Parameter t
5. Graph zeichnen
6. Hardcopy für Schneider-Printer NLQ 401
7. Funktionswerte abfragen: Es können die Funktionswerte für beliebige X-Werte berechnet werden. Rücksprung ins Menü durch Eingabe von »m«.
8. Ende: Programm wird verlassen, ist aber durch »RUN« wieder zu starten.

Man muß auf eine computergerechte Funktionseingabe achten. Wurde eine Funktion fehlerhaft eingegeben (zum Beispiel 5 sin x anstelle von 5*sin(x)), so ist sie nicht definiert und es erfolgt kein Ausdruck.

Beim Menüpunkt »Hardcopy« testet das Programm, ob ein Drucker angeschlossen und einsatzbereit ist, bevor der Hardcopy-RSX-Befehl in einer Endlos-Schleife darauf wartet. ON-ERROR-Sprünge verhindern Fehlermeldungen bei Polen oder nicht definierten Stellen von Funktionen. Die Graphen werden durchgezeichnet und nicht aus Einzelpunkten zusammengesetzt, damit auch bei extremen Steigungen keine Löcher im Schaubild entstehen. Dadurch, daß der Graph, wenn er aus dem Bildschirm hinausläuft, am entsprechenden Bildrand als Gerade weiter gezeichnet wird, lassen sich Sprünge von minus unendlich nach plus unendlich erkennen und somit von nicht definierten Wertebereichen unterscheiden.

Von der Programmierung her am interessantesten ist das Einfügen der Funktion in das Listing. Sie wird als Stringvariable f$ eingegeben und dann in Zeile 220 als Programmzeile 240 auf die kleine ENTER-Taste gelegt. In Zeile 230 wird dann der Tastaturpuffer manipuliert, ein Druck auf diese Taste simuliert und das Programm abgebrochen. Nach dem Abbruch liest das Betriebssystem den Tastaturpuffer aus und startet das Programm wieder in Zeile 240. Der Benutzer merkt von all dem nichts, da die entsprechenden Systemmeldungen durch Gleichsetzen von Vorder- und Hintergrundfarbe unsichtbar bleiben.

Ein Hinweis zum Eintippen: Wenn durch »RENUM« die Zeilennummern verändert werden, müssen in Zeile 220 bei der Belegung der Funktionstasten die entsprechenden Zeilennummern von Hand geändert werden.

(Timo Breidenstein/ja)



Daten parat



Wozu hat man nun einen Computer, wenn man immer noch alles Mögliche auswendig behalten soll? Unsere universelle Dateiverwaltung nimmt Ihnen einiges ab.

Dieses Basic-Programm ist universell einsetzbar und eignet sich sehr gut zum Verwalten kleinerer Datenbestände.

Beachten Sie dabei folgende Punkte:

- Die Auswahl der Funktionen aus den Menüs erfolgt durch Drücken der entsprechenden Zifferntaste; »ENTER« ist nicht erforderlich.
- Geben Sie keine Buchstaben ein, wenn numerische Werte erwartet werden und umgekehrt.
- Die Eingabe eines Punktes ».« dient bei den meisten Funktionen dazu, diese wieder zu verlassen.
- Sollte das Programm trotzdem einmal abstürzen, kann es in den meisten Fällen ohne Datenverlust durch Eingabe von »GOTO 30« wieder gestartet werden.

Nach dem Programmstart geben Sie das Datum ein, und zwar bitte immer in der verlangten Form, da sonst als Datum 31.12.99 erscheint. Danach wird ein Kennwort angefordert, damit Ihre Daten vor unberechtigtem Zugriff geschützt sind. Das Kennwort steht in Zeile 1230 des Listings und Sie können es gegen Ihr eigenes austauschen. Wurde es richtig eingegeben, erscheint das Hauptmenü, ansonsten bricht das Programm ab.

Hauptmenü (Zeilen 300 bis 430):

Von hier aus erreichen Sie folgende Unterprogramme:

-1- Datei definieren
-2- Datei eingeben
-3- Datei pflegen
-4- Datei ausgeben
-5- Datei abspeichern
-6- Datei laden
-7- Programmende

Datei definieren (Zeilen 2000 bis 2240):

Bevor Sie eine neue Datei definieren, überprüft das Programm, ob nicht schon eine Datei im Speicher existiert. Sollte dies der Fall sein, erhalten Sie eine entsprechende Fehlermeldung und das Programm kehrt ins Hauptmenü zurück. Sie haben dann die Möglichkeit, die existente Datei abzuspeichern und zu löschen.

Eine Datei wird definiert, indem Sie die Anzahl von Datensätzen und die Anzahl von Feldern pro Datensatz angeben. Danach ist für jedes Feld eine Bezeichnung (Feldname), ein Typ (alphabetisch oder numerisch) und die Anzahl von Zeichen pro Feld festzulegen.

Stellen Sie an dieser Stelle fest, daß Ihre Angaben einer Änderung bedürfen, so ist zuerst ein Datensatz anzulegen und anschließend die Datei zu löschen.

Daten eingeben (Zeilen 3000 bis 3200):

Sie geben Ihre Datensätze Feld für Feld ein, vorausgesetzt, Sie haben zuvor eine Datei definiert oder geladen.

Auf dem Bildschirm erscheint dazu die nächste Satznummer, gefolgt von den gewählten Feldnamen.

Jede Eingabe ist mit »ENTER« abzuschließen. Das Programm prüft, ob die maximale Anzahl von Zeichen je Feld eingehalten wurde. Je nach Beantwortung der Frage »Eingabe in Ordnung (j/n):« können Sie korrigieren oder mit dem nächsten Datensatz fortfahren.

Um die Funktion zu verlassen, geben Sie beim nächsten Satz im ersten Feld einen Punkt ein.

Die laufende Satznummer wird automatisch immer um eins erhöht, bis die in der Definition festgelegte maximale Anzahl von Sätzen erreicht ist.

Die Berechnung des freien Speicherplatzes kann unter Umständen einige Sekunden dauern.

Datei pflegen (Zeile 4000 bis 4860):

»Datei pflegen« bietet:

-1- Satz ändern
-2- Satz löschen
-3- Datei löschen
-4- Datei sortieren
-5- Haupt-Menü

Durch Drücken der entsprechenden Zifferntaste wählen Sie die gewünschte Funktion.

Satz ändern (Zeilen 4100 bis 4300):

Für die Änderung geben Sie die Satznummer ein. Dieser wird angezeigt und Sie können mit »Cursor down« in das zu ändernde Feld springen. »ENTER« löscht den alten Inhalt und Sie können den neuen eingeben. Mit »CLR« verlassen Sie die Funktion. Wollen Sie weitere Sätze ändern, überspringen Sie mit »Cursor down« das letzte Feld und geben eine weitere Satznummer ein.

Satz löschen (Zeilen 4350 bis 4460 und 8000 bis 8070):

Nach Eingabe einer Satznummer erscheint der Inhalt des Satzes. Danach entscheiden Sie, ob der Satz gelöscht werden soll.

Beim Löschen rückt eine Unterroutine die nachfolgenden Sätze vor und vermindert die höchste zur Zeit vergebene Satznummer um eins.

Datei löschen (Zeilen 4500 bis 4590):

Diese Funktion löscht die Datei im Speicher. Sie können anschließend eine neue Datei definieren oder von Kassette oder Diskette laden.

Sortieren (Zeilen 4600 bis 4860):

Die Datei kann jeweils nur nach einem der angezeigten Schlüssel sortiert werden. Ein Punkt als Schlüssel bewirkt das Verlassen des Programmteils.

Ausgabe (Zeilen 5000 bis 5970):

In diesem Untermenü wählen Sie zwischen:

-1- Einzelne Sätze
-2- Ausgewählte Sätze
-3- Alle Sätze
-4- Haupt-Menü

Einzelne Sätze (Zeilen 5200 bis 5350):

Eine Anzeige von einzelnen Sätzen erfolgt, nachdem Sie die gewünschte Satznummer eingegeben haben.

Die Satznummer ist nicht Bestandteil der Datei, sondern ergibt sich aus der Reihenfolge der Sätze im Speicher.

Das bedeutet, daß sich der Inhalt einer bestimmten Satznummer zu verschiedenen Zeiten (zum Beispiel nach dem Sortieren) unterscheiden kann.

Mit den Cursor-Tasten können Sie rückwärts und vorwärts blättern oder mit »ENTER« abbrechen, um eine neue Satznummer einzugeben. Ein Punkt als Satznummer eingegeben bringt Sie zurück ins Ausgabe-Menü.

Ausgewählte Sätze (Zeilen 5500 bis 5730):

Dieser Programmteil bietet Ihnen eine umfangreiche Suchfunktion innerhalb der Datei. Dazu erscheint im Window 4 unten links ein Fragezeichen. Sie können jetzt den Suchbegriff eingeben. Ist ein Suchbegriff in mehr als 15 Datensätzen enthalten, hält die Ausgabe an. An dieser Stelle können Sie entweder mit »Cursor down« weitere Datensätze anzeigen lassen, oder mit »ENTER« die Ausgabe abbrechen, um einen neuen Suchbegriff auszuwählen. Ein Punkt als Suchbegriff bricht ab und Sie kehren ins Ausgabe-Menü zurück.

Beispiele

»?: Haus«	Findet alle Sätze, deren erste vier Zeichen bei alphabetischen Feldern »Haus« enthalten. Auch »Hausfrau« fällt beispielsweise darunter. Lassen



Sie aber auf »Haus« einen Leerschritt folgen, grenzen Sie so die Suche ein.

»?: 10.00« — Zeigt alle Sätze, deren letzte fünf Zeichen bei numerischen Feldern »10.00« enthalten. Auch »210.00« wird ausgegeben. Hier können Sie vor dem Suchbegriff einen Leerschritt einfügen, um die Suche einzugrenzen.

»?: Preis>100.00« — Findet alle Sätze, deren Feld Preis einen Wert größer oder gleich 100.00 hat. Auch 32100.00 würde angezeigt, wenn nicht nach dem > ein Leerschritt folgt (Preis> 100.00).

»?: Bezeichnung<M« — Findet alle Sätze, deren Feld Bezeichnung mit einem »M« oder einem Buchstaben beginnt, der im Alphabet vor dem M steht.

Achten Sie darauf, daß der Suchbegriff nicht länger als die erlaubte Anzahl von Zeichen des Feldes ist.

Sämtliche Sätze (Zeilen 5800 bis 5970):

Alle Sätze der Datei werden auf dem Bildschirm ausgegeben, wobei nach jeweils 15 Sätzen die Ausgabe anhält. Sie können dann mit »Cursor down« weitermachen oder mit »ENTER« abbrechen (siehe Window #3 rechts unten).

Datei speichern (Zeilen 6000 bis 6240):

Aufgezeichnet werden:

- Dateiname
- Anzahl Sätze, Anzahl Felder
- Feldnamen, Feldtypen, Anzahl der Zeichen pro Feld
- höchste belegte Satznummer+1
- alle eingegebenen Feldinhalte

Datei laden (Zeilen 7000 bis 7350):

Es darf keine Datei im Speicher existieren, soll diese Funktion ausgeführt werden. Eine eventuell existente Datei ist gegebenenfalls abzuspeichern und im Arbeitsspeicher zu löschen.

Die Eingabe des Dateinamens bestimmt die zu ladende Datei.

Programmende (Zeilen 10000 bis 10060):

Bevor das Programm endgültig endet, erfolgt nochmals die Frage, ob Sie das auch wirklich beabsichtigen. Vielleicht haben Sie vergessen, die Datei vorher abzuspeichern.

Beim Beenden wird der Bildschrim gelöscht und die Ready-Meldung erscheint in Zeile 1 am Bildschrim.

Sie könnten jetzt das Programm wieder aktivieren, sogar ohne Datenverlust, durch Eingabe von »GOTO 30«.

(Josef Tiefenböck/ja)

```
1 REM J.Tiefenboeck                          [A858]
2 REM Oberer Weg 11                          [EEF4]
3 REM 8904 Staetzling                        [37E2]
4 REM tel. 0821/7102701                      [808E]
5 '                                          [ACF8]
6 '                                          [CCFA]
10 REM *** INITIALIZE ***                    [E066]
20 MODE 2:GOSUB 1000                         [7328]
30 WINDOW 1,80,4,22                          [D660]
40 WINDOW #2,1,80,1,3                        [B8FC]
50 WINDOW #3,41,80,23,25                     [0D38]
55 WINDOW #4,1,40,23,25                      [70D4]
60 BORDER 6                                  [A634]
70 OPENOUT "dummy":MEMORY HIMEM-1:CLOSEO
   UT                                        [95D4]
90 GOTO 300                                  [61EA]
100 REM *** UMRANDUNGEN ***                  [4E6A]
110 LOCATE #2,28,2: PRINT #2,"H A U P T
    M E N U E";                              [20EC]
120 LOCATE #2,64,2: PRINT #2,TT$;"/";MM$
    ;"/";JJ$;:RETURN                         [368A]
200 LOCATE #2,1,1: PRINT #2,CHR$(150);ST
    RING$(78,CHR$(154));CHR$(156);           [D7F6]
205 LOCATE #2,1,2: PRINT #2,CHR$(149);ST
    RING$(78,CHR$(32));CHR$(149);            [BCAC]
210 LOCATE #2,1,3: PRINT #2,CHR$(147);ST
    RING$(78,CHR$(154));CHR$(153);:RETUR
    N                                        [DD36]
220 LOCATE #3,1,1: PRINT #3,CHR$(150);ST
    RING$(38,CHR$(154));CHR$(156);           [35F6]
230 LOCATE #3,1,2: PRINT #3,CHR$(149);ST
    RING$(38,CHR$(32));CHR$(149);            [09A4]
240 LOCATE #3,1,3: PRINT #3,CHR$(147);ST
    RING$(38,CHR$(154));CHR$(153);:RETUR
    N                                        [3638]
250 LOCATE #4,1,1: PRINT #4,CHR$(150);ST
    RING$(38,CHR$(154));CHR$(156);           [2400]
260 LOCATE #4,1,2: PRINT #4,CHR$(149);ST
    RING$(38,CHR$(32));CHR$(149);            [D4AE]
270 LOCATE #4,1,3: PRINT #4,CHR$(147);ST
    RING$(38,CHR$(154));CHR$(153);:RETUR
    N                                        [FF42]
300 REM *** HAUPTMENUE ***                   [38DE]
305 CLS                                      [8334]
310 GOSUB 200:GOSUB 110:GOSUB 220:GOSUB
    250                                      [6362]
320 LOCATE 20,3: PRINT "- 1 -{6 SPACE}Da
    tei definieren"                          [B088]
330 LOCATE 20,5: PRINT "- 2 -{6 SPACE}Da
    ten eingeben"                            [C1E2]
340 LOCATE 20,7: PRINT "- 3 -{6 SPACE}Da
    tei pflegen"                             [C128]
350 LOCATE 20,9: PRINT "- 4 -{6 SPACE}Da
    tei ausgeben"                            [8A02]
360 LOCATE 20,11: PRINT "- 5 -{6 SPACE}D
    atei abspeichern"                        [85CC]
370 LOCATE 20,13: PRINT "- 6 -{6 SPACE}D
    atei laden"                              [08D4]
380 LOCATE 20,15: PRINT "- 7 -{6 SPACE}P
    ROGRAMM ENDE"                            [1508]
390 LOCATE 20,16:PRINT STRING$(32,CHR$(1
    54));                                    [62E8]
400 LOCATE 20,17:PRINT "Bitte waehlen Si
    e:{2 SPACE}";                            [E83E]
405 hw$=INKEY$:IF hw$="" THEN 405            [BA24]
410 hw=VAL(hw$):IF hw<1 OR hw>7 GOTO 405
                                             [D108]
420 ON hw GOSUB 2000,3000,4000,5000,6000
    ,7000,10000                              [224C]
430 GOTO 300                                 [E546]
700 REM *** FEHLER BEHANDLUNG ***            [CC4A]
730 LOCATE #4,2,2:PRINT #4,SPACE$(36);:P
    RINT CHR$(7);                            [FF7E]
740 LOCATE #4,3,2:PRINT #4,fehler$;          [C0CC]
750 FOR z%=1 TO 2000:NEXT z%:GOSUB 250:R
    ETURN                                    [5670]
800 REM *** Freier Speicher ***              [1D50]
810 LOCATE #2,2,2:PRINT#2, FRE("");:RETU
    RN                                       [D8A2]
1000 REM *** ANMELDUNG ***                   [4678]
1010 PRINT STRING$(80,CHR$(208));            [BF2C]
1020 LOCATE 20,3:PRINT "D A T E N B A N
     K{6 SPACE}C P C{2 SPACE}4 6 4"          [AB46]
1030 PRINT STRING$(80,CHR$(210));            [5B22]
1040 LOCATE 20,7:PRINT CHR$(164);"{2 SPA
     CE}by jtb{2 SPACE}1985"                 [E5CE]
1050 LOCATE 20,10:INPUT "DATUM{3 SPACE}(
     TT/MM/JJ):{2 SPACE}",dat$               [A47C]
1070 TT$=LEFT$(dat$,2):TT=VAL(TT$)           [51D6]
1080 IF TT<1 OR TT>31 THEN LOCATE 1,10:P
     RINT CHR$(20);:GOTO 1050                [6018]
1090 MM$=MID$(dat$,4,2):MM=VAL(MM$)          [A5A4]
1100 IF MM<1 OR MM>12 THEN LOCATE 1,10:P
     RINT CHR$(20);:GOTO 1050                [F6D0]
1110 JJ$=RIGHT$(dat$,2):JJ=VAL(JJ$)          [97FA]
1120 IF JJ<85 OR JJ>99 THEN LOCATE 1,10:
     PRINT CHR$(20);:GOTO 1050               [3352]
1200 LOCATE 20,18:PRINT"Kennwort:{2 SPAC
     E}";                                    [5894]
1210 CALL &BB57:INPUT "",kw$:CALL &BB54:
     LOCATE 30,18                            [7252]
1220 FOR k=1 TO LEN(kw$):PRINT "*";:NEXT
      k                                      [72E4]
1230 IF kw$<>"happy" THEN END ELSE RETUR
     N                                       [2614]
2000 REM *** Datei definieren ***            [F784]
2010 CLS                                     [198A]
2020 GOSUB 200:GOSUB 250:GOSUB 220           [8EEA]
2023 LOCATE #2,28,2:PRINT #2,"D E F I N
     I T I O N";                             [19F0]
2025 IF lr>0 THEN fehler$="Datei schon d
     efiniert": GOSUB 700:GOTO 300           [A342]
2030 LOCATE 4,4:INPUT "Anzahl Saetze{2 S
     PACE}(insgesamt max. 250):{3 SPACE}
     ",ar                                    [FE52]
2040 IF ar>250 THEN 2030                     [28E2]
2045 LOCATE 4,10:PRINT CHR$(24)+" Summe
     aller Zeichen inklusive Leerzeichen
      <80 ! "+CHR$(24)                       [58BE]
2050 LOCATE 4,6:INPUT "Anzahl Felder{2 S
     PACE}(eine Zeile,max. 10):{4 SPACE}
     ",af                                    [88D8]
2060 IF af>10 THEN LOCATE 4,10:PRINT CHR
     $(24)+"{8 SPACE}maximal 10 Felder !
     {41 SPACE}"+CHR$(24):GOTO 2050          [0E1A]
2070 LOCATE #3,3,2:PRINT #3,"Saetze:";ar
     ;"{2 SPACE}Felder:";af;                 [A07A]
2080 CLS:DIM f$(ar,af):lr=1:sum=0:sumbz=
     0                                       [E4F6]
2090 FOR i= 1 TO af                          [7C26]
2092 LOCATE 4,12:PRINT CHR$(24)+" Summe
     aller Zeichen inklusive Leerzeichen
```

Listing. Datenverwaltung, wie sie komfortabler nicht sein könnte

```
     <80 ! "+CHR$(24)                          [D3C6]
2095 LOCATE 4,3:PRINT CHR$(18);"Feld ";i
     ;                                         [9EEC]
2100 LOCATE 4,5: INPUT "Bezeichnung:{8 S
     PACE}",fb$(i)                             [FE38]
2110 LOCATE 4,7: INPUT "Type (Alpha/Num.
     ):{2 SPACE}",ft$(i)                       [A30A]
2120 LOCATE 4,9: INPUT "Anzahl Zeichen:{
     5 SPACE}",fc(i)                           [35E2]
2125 sum= sum+fc(i): sumbz=sumbz+LEN(fb$
     (i))                                      [2332]
2130 LOCATE 4,12: PRINT "Summe der Einga
     ben bisher ";sum;" Zeichen.Maximal
     inklusiv Leerzeichen <80!"                [1630]
2135 LOCATE #4,2,2:PRINT #4,"Noch frei :
     ";(80-SUM)-((fc(i)-LEN(fb$(i))+hva
     r2));" Zeichen"                           [3154]
2140 LOCATE 4,15: INPUT "Eingabe in Ordn
     ung (j/n): ",a$                           [F8EA]
2150 IF UPPER$(a$)<>"J" THEN 2155 ELSE 2
     160                                       [2226]
2155 LOCATE 1,3:PRINT CHR$(20):sum=sum-f
     c(i):sumbz=sumbz-LEN(fb$(i)):GOTO 2
     095                                       [9500]
2160 LOCATE 1,3:PRINT CHR$(20);                [714E]
2170 NEXT i                                    [AA64]
2180 IF sum>(80-af) OR sumbz>(80-af) THE
     N 2190 ELSE 2210                          [F580]
2190 ERASE f$,fb$,ft$,fc:lr=0                  [9E84]
2200 fehler$="Summe Zeichen > Zeilenlaen
     ge": GOSUB 700: GOTO 2000                 [7186]
2210 hvar1=0:FOR i=1 TO af:hvar1=hvar1+f
     c(i):NEXT                                 [56B0]
2220 hvar2=FIX((80-hvar1)/af)                  [0CF8]
2230 IF hvar2<1 THEN hvar2=1 ELSE IF hva
     r2>10 THEN hvar2=10                       [4D60]
2240 RETURN                                    [8F90]
3000 REM *** E I N G A B E ***                 [B59C]
3010 CLS                                       [0A8C]
3020 GOSUB 200:GOSUB 220:GOSUB 250             [1BEC]
3023 LOCATE #2,28,2: PRINT #2,"E I N G A
      B E ";                                   [32B6]
3025 IF lr=0 THEN fehler$="Keine Daten v
     orhanden !": GOSUB 700: RETURN            [9AAE]
3027 GOSUB 800                                 [3348]
3030 LOCATE #3,3,2: PRINT #3,"Saetze ";a
     r;"{2 SPACE}Felder ";af;                  [C84C]
3040 IF lr>0 THEN 3050                         [DD32]
3045 lr=1                                      [8F70]
3050 WHILE lr<=ar                              [1A56]
3055 LOCATE 2,18:PRINT "Verlassen mit >.
     < "                                       [CC78]
3060 LOCATE 2,2:PRINT CHR$(18);"SATZ ";l
     r                                         [1ADA]
3070 FOR i= 1 TO af                            [C624]
3080 LOCATE 2,i+3:PRINT CHR$(18);fb$(i);
     ":"                                       [C648]
3090 NEXT i                                    [0768]
3100 FOR i= 1 TO af                            [C518]
3110 LOCATE 5+LEN(FB$(I)),i+3:PRINT CHR$
     (18);:INPUT "",e$                         [698C]
3115 IF LEN(e$)>fc(i) THEN GOSUB 3200: G
     OTO 3110                                  [99BE]
3120 IF UPPER$(e$)<>"." THEN 3122 ELSE G
     OSUB 800:RETURN                           [F4FA]
3122 IF UPPER$(ft$(i))="N" THEN 3125 ELS
     E 3127                                    [FE1E]
3125 f$(lr,i)=SPACE$(fc(i)-LEN(e$))+e$:G
     OTO 3130                                  [DD9C]
3127 f$(lr,i)=e$                               [CC02]
3130 NEXT i                                    [905E]
3140 LOCATE 2,18: INPUT "Eingabe in Ordn
     ung (j/n):{2 SPACE}",a$                   [B12E]
3150 IF UPPER$(a$)="J" THEN 3160 ELSE LO
     CATE 1,18:PRINT CHR$(18);:GOTO 3060
                                               [2BC4]
3160 lr=lr+1                                   [D77E]
3170 LOCATE 1,18:PRINT CHR$(18);               [74CC]
3180 WEND                                      [0934]
3190 GOSUB 800:RETURN                          [6A7E]
3200 fehler$=fb$(i)+" zu lang":GOSUB 700
     :RETURN                                   [315A]
4000 REM *** DATEI PFLEGE ***                  [04BC]
4010 CLS                                       [2B8E]
4020 GOSUB 200:GOSUB 250: GOSUB 220            [062E]
4023 LOCATE #2,28,2:PRINT #2,"P F L E G
     E";                                       [7288]
4025 IF lr=0 THEN fehler$="Keine Daten v
     orhanden !": GOSUB 700: RETURN            [89B0]
4030 LOCATE 20,4: PRINT "- 1 -{7 SPACE}S
     atz aendern"                              [0E2C]
4031 LOCATE 20,6: PRINT "- 2 -{7 SPACE}S
     atz loeschen"                             [821C]
4032 LOCATE 20,8: PRINT "- 3 -{7 SPACE}D
     atei loeschen"                            [6FAE]
4033 LOCATE 20,10: PRINT "- 4 -{7 SPACE}
     Datei sortieren"                          [E218]
4034 LOCATE 20,12: PRINT "- 5 -{7 SPACE}
     Haupt-Menue"                              [CFAE]
4035 LOCATE 20,16: PRINT "Bitte waehlen
     sie: ";                                   [C5EC]
4036 uw$=INKEY$: IF uw$="" THEN 4036           [8C68]
4037 IF uw$="1" THEN 4100                      [FF26]
4038 IF uw$="2" THEN 4350                      [5838]
4039 IF uw$="3" THEN 4500                      [F736]
4040 IF uw$="4" THEN 4600                      [1F2A]
4041 IF uw$="5" THEN RETURN ELSE 4036          [F1C6]
4100 REM *** Record aendern ***                [5702]
4110 CLS                                       [3590]
4115 LOCATE 2,18:PRINT "Verlassen mit >.
     < "                                       [5374]
4120 GOSUB 250:LOCATE #4,2,2:INPUT #4,"S
     atz - Nummer: ",sn$                       [3B04]
4130 IF sn$="." THEN 4310                      [4004]
4140 sn=VAL(sn$): IF sn=0 THEN GOTO 4120
                                               [0C4C]
4150 IF sn>ar THEN fehler$="Satznummer z
     u gross": GOSUB 700: GOTO 4100            [3862]
4160 IF sn<1 THEN sn=1 ELSE IF sn>lr-1 T
     HEN sn=lr-1                               [D0C4]
4170 LOCATE 2,3: PRINT "Satz : ";sn            [24B0]
4180 FOR i=1 TO af                             [86EA]
4190 LOCATE 2,4+i:PRINT CHR$(18);fb$(i);
     TAB(20);f$(sn,i);                         [A5F6]
4200 NEXT i                                    [9B5C]
4210 i=1                                       [E07C]
4220 LOCATE #3,2,2: PRINT #3,CHR$(241);"
     =weiter{2 SPACE}<ENTER>=KORR <CLR>=
     ENDE";                                    [70A4]
4225 CALL &BB9F                                [69A4]
4230 LOCATE 20,4+i:CALL &BB81                  [FF40]
4240 a$=INKEY$: IF a$="" THEN 4240             [4330]
4250 IF a$=CHR$(241) THEN i=i+1: GOTO 42
     90                                        [E142]
4262 IF a$=CHR$(13) THEN 4265 ELSE IF a$
     =CHR$(16) THEN 4310                       [EB3E]
4265 LOCATE 20,4+i:PRINT CHR$(18);:INPUT
      "",e$                                    [B7BE]
4270 IF LEN(e$)>fc(i) THEN GOSUB 4300: G
     OTO 4230                                  [4ED0]
4275 f$(sn,i)=e$                               [0512]
4280 IF UPPER$(ft$(i))="N" THEN f$(sn,i)
     =SPACE$(fc(i)-LEN(e$))+e$                 [1740]
4285 LOCATE 20,4+i:PRINT CHR$(18);f$(sn,
     i);:i=i+1                                 [875E]
4290 IF i>af THEN PRINT CHR$(7): GOTO 41
     20                                        [0056]
4295 GOTO 4230                                 [B62C]
4300 fehler$=fb$(i)+" zu lang": GOSUB 70
     0: RETURN                                 [8CDE]
4310 CALL &BB9F:CALL &BB84:GOTO 4000           [4E60]
4350 REM *** Records loeschen ***              [D0DE]
4360 CLS                                       [189E]
4365 LOCATE 2,18:PRINT "Verlassen mit >.
     < "                                       [2D82]
4370 LOCATE #4,2,2: INPUT #4,"Satz - Num
     mer: ",sn$                                [1970]
4380 IF sn$="." THEN 4000                      [E50A]
4390 sn=VAL(sn$): IF sn=0 THEN 4370            [1BB6]
4400 IF sn>lr-1 THEN fehler$="Satznummer
      zu gross !":GOSUB 700:GOTO 4350          [6940]
4410 IF sn<1 THEN sn=1                         [4548]
4415 LOCATE 2,3: PRINT "Satz : ";sn            [9CB4]
4420 FOR i=1 TO af:LOCATE 2,4+i: PRINT f
     b$(i);TAB(20);f$(sn,i);:NEXT i            [63CA]
4430 LOCATE #4,2,2: INPUT #4,"Loeschen (
     j/n):{2 SPACE}",a$                        [39F6]
4440 IF UPPER$(a$)="J" THEN f$(sn,1)="*"
     :GOSUB 8000                               [AA82]
4450 GOSUB 250                                 [1448]
4460 GOTO 4350                                 [4526]
4500 REM *** Datei loeschen ***                [9D02]
4510 CLS                                       [0D98]
4520 GOSUB 250: GOSUB 220                      [B960]
4530 LOCATE #3,2,2: PRINT #3,"Datei:{2 S
     PACE}";n$;                                [3718]
4540 LOCATE #4,2,2: INPUT #4,"Loeschen (
     j/n):{2 SPACE}",a$                        [3AFA]
4550 IF UPPER$(a$)="J" THEN 4560 ELSE 45
     90                                        [A1CE]
4560 ERASE f$,fb$,ft$,fc:GOSUB 800             [D664]
4580 lr=0:FOR i=1 TO 1500:NEXT                 [05EC]
4590 GOTO 4000                                 [611E]
4600 REM *** Sortieren  SHELL-SORT ***         [8D24]
4610 CLS: GOSUB 250: GOSUB 220                 [F6D8]
4612 LOCATE 1,2:FOR i=1 TO af:PRINT fb$(
     i);SPC(fc(i)-LEN(fb$(i))+hvar2);:NE
     XT                                        [CCBA]
4615 LOCATE 2,18:PRINT "Verlassen mit >.
     < "                                       [287E]
4620 LOCATE #4,2,2: INPUT #4,"Schluessel
     : ",s$                                    [F0C0]
4630 IF s$="." THEN 4000                       [5C2A]
4640 FOR i=1 TO af                             [88EC]
4650 ss=0: IF s$=fb$(i) THEN ss=i: GOTO
     4670                                      [1E88]
4660 NEXT i                                    [CF70]
4670 IF ss=0 THEN fehler$="Falscher Schl
     uessel !": GOSUB 700: GOTO 4600           [805E]
4675 LOCATE #3,2,2: PRINT #3, "Bitte war
     ten !"                                    [236E]
4676 DIM h$(ar,af)                             [B428]
```

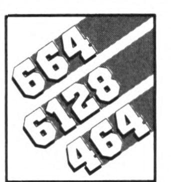
```
4677 FOR i=1 TO lr:FOR j=1 TO af:h$(i,j)
     =f$(i,j):f$(i,j)="":NEXT:NEXT          [C51C]
4678 GOSUB 800                              [7362]
4680 m=lr-1                                 [FFB0]
4690 m=INT(m/2)                             [1E4E]
4700 IF m=0 THEN 4830                       [AA64]
4710 j=1                                    [D288]
4720 k=(lr-1)-m                             [D778]
4730 n=j                                    [8B06]
4740 l=n+m                                  [533C]
4750 IF h$(n,ss) < h$(l,ss) THEN 4800       [B01C]
4760 FOR i=1 TO af                          [8CF2]
4761 h$(0,i)=h$(n,i): h$(n,i)=h$(l,i): h
     $(l,i)=h$(0,i)                         [D83A]
4762 NEXT i                                 [A876]
4770 n=n-m                                  [504A]
4780 IF n<1 THEN 4800                       [B270]
4790 GOTO 4740                              [AD38]
4800 j=j+1                                  [28B2]
4810 IF j <=k THEN 4730                     [CD8E]
4820 GOTO 4690                              [6F34]
4830 i=0:j=0:k=0:l=0:m=0:n=0                [8648]
4840 FOR i=1 TO lr:FOR j=1 TO af:f$(i,j)
     =h$(i,j):NEXT:NEXT                     [B9E2]
4845 ERASE h$:GOSUB 800                     [2B06]
4850 GOSUB 250: LOCATE #4,2,2: PRINT #4,
     "Datei ist sortiert !"                 [5FDE]
4860 FOR i=1 TO 1500: NEXT i: GOSUB 250:
      GOSUB 220:GOTO 4600                   [5B2E]
5000 REM *** AUSGABE ***                    [A33A]
5010 CLS                                    [1C90]
5020 GOSUB 200:GOSUB 250: GOSUB 220         [BA30]
5023 LOCATE #2,28,2: PRINT #2,"A U S G A
      B E";                                 [5E94]
5030 IF lr=0 THEN fehler$="Keine Daten v
     orhanden !": GOSUB 700: RETURN         [DDAA]
5050 LOCATE 20,4: PRINT "- 1 -{5 SPACE}E
     inzelne Saetze"                        [4200]
5060 LOCATE 20,6: PRINT "- 2 -{5 SPACE}A
     usgewaehlte Saetze"                    [AC52]
5070 LOCATE 20,8: PRINT "- 3 -{5 SPACE}A
     lle Saetze"                            [9D98]
5080 LOCATE 20,10: PRINT "- 4 -{5 SPACE}
     Haupt-Menue"                           [1D2C]
5090 LOCATE 20,14: PRINT "Bitte waehlen
     sie: ";                                [AEEC]
5100 uw$=INKEY$: IF uw$="" THEN 5100        [6A4C]
5110 IF uw$="1" THEN 5200                   [C41C]
5120 IF uw$="2" THEN 5500                   [D926]
5130 IF uw$="3" THEN 5800                   [3A30]
5140 IF uw$="4" THEN RETURN ELSE 5100       [06B8]
5200 REM *** einzel Record ***              [0C5A]
5210 CLS                                    [3494]
5220 LOCATE 2,18:PRINT "Verlassen mit >.
     < "                                    [2F70]
5230 LOCATE #4,2,2: INPUT #4,"Satz - Num
     mer: ",sn$                             [F868]
5240 IF sn$="." THEN GOTO 5000              [8BB6]
5250 sn=VAL(sn$): IF sn=0 THEN GOSUB 250
     : GOTO 5230                            [F77A]
5260 CLS                                    [939E]
5265 IF sn>ar THEN fehler$="Satznummer z
     u gross": GOSUB 700: GOTO 5200         [CF76]
5267 IF sn<1 THEN sn=1                      [2B5E]
5268 IF sn>(lr-1) THEN sn=lr-1              [2232]
5270 LOCATE 2,3: PRINT "Record: ";sn        [B9EE]
5280 FOR i=1 TO af                          [85EE]
5290 LOCATE 2,4+i: PRINT fb$(i);TAB(20);
     f$(sn,i);                              [FA4E]
5300 NEXT i                                 [D460]
5310 LOCATE #3,2,2: PRINT #3,CHR$(241);"
     =vor ";CHR$(240);" =zurueck";          [91B6]
5315 LOCATE #3,24,2: PRINT #3,"<ENTER>=E
     NDE";                                  [EDCA]
5320 a$=INKEY$: IF a$="" THEN 5320          [3130]
5330 IF a$=CHR$(13) THEN GOSUB 250: GOTO
      5200                                  [7A64]
5340 IF a$=CHR$(241) THEN sn=sn+1: GOTO
     5260                                   [1220]
5350 IF a$=CHR$(240) THEN sn=sn-1: GOTO
     5260 ELSE 5320                         [D98A]
5500 REM *** Ausgewaehlte Records ***       [9EF6]
5510 CLS:GOSUB 250                          [0F7C]
5515 LOCATE 2,10:PRINT "Verlassen mit >.
     < "                                    [EF6E]
5518  LOCATE 1,2:FOR i=1 TO af:PRINT fb$
     (i);SPC(fc(i)-LEN(fb$(i))+hvar2);:N
     EXT                                    [2406]
5520 LOCATE #4,2,2:PRINT #4,"Bitte Suchk
     riterium waehlen ":LOCATE #3,2,2:PR
     INT #3, "{29 SPACE}":LOCATE #3,2,2:
     INPUT #3," ?: ",such$                  [FFA2]
5525 IF such$="." THEN 5000                 [ECB4]
5530 LOCATE 1,2: PRINT CHR$(24);            [3E9C]
5540 FOR i=1 TO af:PRINT fb$(i);SPC(fc(i
     )-LEN(fb$(i))+hvar2);:NEXT             [247A]
5550 PRINT CHR$(24);:i=1:j=1:z=1            [BEA8]
5560 LOCATE 1,3+z:PRINT CHR$(20);           [E2A6]
5561 LOCATE 1,3+z                           [DEFC]
5562 a=INSTR(such$,">"):IF a>0 THEN 5570
                                            [ADDC]
5563 a=INSTR(such$,"<"):IF a>0 THEN 5580
                                            [02DC]
5564 a=INSTR(such$,"="):IF a>0 THEN 5590
                                            [FEE2]
5565 IF LEFT$(f$(i,j),LEN(such$))=such$
     THEN 5600                              [733E]
5566 IF RIGHT$(f$(i,j),LEN(such$))=such$
      THEN 5600                             [FEE6]
5567 IF j<af THEN j=j+1:GOTO 5565           [CAAE]
5568 IF i<lr-1 THEN i=i+1:j=1:GOTO 5565
     ELSE 5700                              [3D22]
5569 fehler$="Falsche Eingabe":GOSUB 700
     :GOTO 5500                             [A7C6]
5570 FOR x=1 TO af                          [E910]
5571 IF LEFT$(such$,a-1)=fb$(x) THEN 557
     4                                      [E272]
5572 NEXT                                   [1464]
5573 fehler$="Falsche Feldbezeichnung":G
     OSUB 700:GOTO 5710                     [2286]
5574 IF UPPER$(ft$(x))="N" THEN 5576 ELS
     E 5575                                 [5980]
5575 IF LEFT$(f$(i,x),LEN(MID$(such$,a+1
     )))>=MID$(such$,a+1)THEN 5600 ELSE
     5577                                   [E0FA]
5576 IF RIGHT$(f$(i,x),LEN(MID$(such$,a+
     1)))>=MID$(such$,a+1) THEN 5600        [3760]
5577 IF i<lr-1 THEN i=i+1:GOTO 5574 ELSE
      5700                                  [FBFE]
5580 FOR x=1 TO af                          [A812]
5581 IF LEFT$(such$,a-1)=fb$(x) THEN 558
     4                                      [0376]
5582 NEXT                                   [FC66]
5583 fehler$="Falsche Feldbezeichnung":G
     OSUB 700:GOTO 5710                     [9388]
5584 IF UPPER$(ft$(x))="N" THEN 5586 ELS
     E 5585                                 [FC86]
5585 IF LEFT$(f$(i,x),LEN(MID$(such$,a+1
     )))<=MID$(such$,a+1)THEN 5600 ELSE
     5587                                   [F1FA]
5586 IF RIGHT$(f$(i,x),LEN(MID$(such$,a+
     1)))<=MID$(such$,a+1) THEN 5600        [FF5E]
5587 IF i<lr-1 THEN i=i+1:GOTO 5584 ELSE
      5700                                  [A102]
5590 FOR x=1 TO af                          [BB14]
5591 IF LEFT$(such$,a-1)=fb$(x) THEN 559
     4                                      [087A]
5592 NEXT                                   [0468]
5593 fehler$="Falsche Feldbezeichnung":G
     OSUB 700:GOTO 5710                     [E48A]
5594 IF UPPER$(ft$(x))="N" THEN 5596 ELS
     E 5595                                 [0F8C]
5595 IF LEFT$(f$(i,x),LEN(MID$(such$,a+1
     )))=MID$(such$,a+1)THEN 5600 ELSE 5
     597                                    [AE86]
5596 IF RIGHT$(f$(i,x),LEN(MID$(such$,a+
     1)))=MID$(such$,a+1) THEN 5600         [3CE8]
5597 IF i<lr-1 THEN i=i+1:GOTO 5594 ELSE
      5700                                  [A606]
5600 FOR p=1 TO af: PRINT f$(i,p);SPC(fc
     (p)-LEN(f$(i,p))+hvar2);:NEXT          [65B8]
5605 IF i<lr-1 THEN 5610 ELSE 5700          [96E0]
5610 IF z<15 THEN z=z+1: i=i+1: j=1: GOT
     O 5561                                 [E05C]
5630 z=1: LOCATE #3,2,2: PRINT #3,CHR$(2
     41)" = weiter{3 SPACE}<ENTER> = END
     E";                                    [E912]
5640 a$=INKEY$: IF a$="" THEN 5640          [8944]
5650 IF a$=CHR$(241) THEN i=i+1:j=1:GOTO
      5560 ELSE IF a$=CHR$(13) THEN 5000
                                            [95BA]
5700 fehler$="Ende der Datei erreicht !"
     : GOSUB 700: GOSUB 220                 [AA76]
5710 LOCATE #3,2,2: PRINT #3,CHR$(241)"
     = Suche{3 SPACE}<ENTER> = MENUE";      [BD18]
5720 a$=INKEY$: IF a$="" THEN 5720          [0940]
5730 IF a$=CHR$(241) THEN 5500 ELSE IF a
     $=CHR$(13) THEN 5000 ELSE 5720         [63FA]
5800 REM *** Alle Saetze ***                [B46E]
5810 CLS                                    [CCA0]
5820 LOCATE 1,2: PRINT CHR$(24);            [65A0]
5830 FOR i=1 TO af:PRINT fb$(i);SPC(fc(i
     )-LEN(fb$(i))+hvar2);:NEXT             [C97E]
5840 PRINT CHR$(24);:i=1:j=1:z=1            [7DAC]
5850 LOCATE 1,3+z:PRINT CHR$(20);           [C7AA]
5860 LOCATE 1,3+z                           [ED00]
5870 FOR j=1 TO af: PRINT f$(i,j);SPC(fc
     (j)-LEN(f$(i,j))+hvar2);:NEXT          [AB9A]
5880 IF i<lr-1 THEN i=i+1 ELSE GOTO 5950
                                            [68E8]
5890 IF z<15 THEN z=z+1: GOTO 5860          [4386]
5900 z=1: LOCATE #3,2,2: PRINT #3,CHR$(2
     41)" = weiter{3 SPACE}<ENTER> = END
     E";                                    [3412]
5910 a$=INKEY$: IF a$="" THEN 5910          [6944]
5920 IF a$=CHR$(241) THEN 5850 ELSE IF a
     $=CHR$(13) THEN 5000 ELSE 5910         [9B0E]
5950 fehler$="Ende der Datei erreicht !"
     : GOSUB 700: GOSUB 220                 [D584]
5960 LOCATE #3,2,2: PRINT #3,"<ENTER> =
     MENUE";                                [5AAA]
5970 a$=INKEY$: IF a$="" THEN 5970 ELSE
     5000                                   [B2B8]
6000 REM *** Datei abspeichern ***          [BB62]
```




```
6010 CLS                                      [2592]
6020 GOSUB 200:GOSUB 250                      [DB16]
6023 LOCATE #2,28,2: PRINT #2,"A B S P E
       I C H E R N";                          [D8EE]
6025 IF lr=0 THEN fehler$="Keine Daten v
     orhanden !": GOSUB 700: RETURN           [F3B4]
6030 LOCATE 1,10: INPUT "Name der Datei:
     ", n$                                    [97D0]
6040 LOCATE 1,15: PRINT "Bitte Cassette
     einlegen. REC und PLAY druecken !"       [A544]
6045 LOCATE 1,16: PRINT "Druecken Sie EN
     TER !";: CALL &BB18                      [9316]
6050 nf$="!"+n$:OPENOUT nf$                   [107C]
6060 PRINT #9,n$                              [B566]
6070 PRINT #9,ar                              [D7EA]
6080 PRINT #9,af                              [B8D4]
6090 FOR i=1 TO af                            [89EE]
6100 PRINT #9,fb$(i)                          [A884]
6110 PRINT #9,ft$(i)                          [5FAA]
6120 PRINT #9,fc(i)                           [7642]
6130 NEXT i                                   [C264]
6140 PRINT #9,lr                              [DFFC]
6150 FOR i=1 TO lr                            [0716]
6160 FOR j=1 TO af                            [AAEC]
6170 PRINT #9,f$(i,j)                         [5EFA]
6180 NEXT j                                   [9A70]
6190 NEXT i                                   [AA70]
6200 CLOSEOUT                                 [6DAC]
6210 LOCATE #4,3,2: PRINT #4,n$;" gespei
     chert !";                                [253E]
6220 FOR z= 1 TO 2000:NEXT z                  [5762]
6230 GOSUB 250                                [3C44]
6240 RETURN                                   [A798]
7000 REM *** DATEI LADEN ***                  [0924]
7010 CLS                                      [3694]
7020 GOSUB 200:GOSUB 250: GOSUB 220           [3234]
7023 LOCATE #2,28,2: PRINT #2,"L A D E N
     ";                                       [90F0]
7030 IF lr>0 THEN fehler$="Datei noch ex
     istent !": GOSUB 700: RETURN             [2C7C]
7040 LOCATE 2,10: INPUT "Name der Datei
     zum Laden: ",n$                          [3196]
7050 LOCATE 2,12: PRINT "Bitte PLAY drue
     cken !";                                 [6F16]
7060 OPENIN"!"+n$                             [3FB0]
7070 INPUT#9,file$                            [CF94]
7072 IF file$<>n$THEN CLOSEIN:fehler$=fi
     le$+"{2 SPACE}gelesen !":GOSUB 700:
     GOTO 7000                                [DC7C]
7075 LOCATE #3,3,2: PRINT #3,"Datei: ";f
     ile$                                     [8ED6]
7080 INPUT #9,ar                              [8AF4]
7090 INPUT #9,af                              [8BDE]
7110 DIM f$(ar,af)                            [BF08]
7120 FOR i=1 TO af                            [8FE4]
7130 INPUT #9,fb$(i)                          [5192]
7140 INPUT #9,ft$(i)                          [A8B8]
7150 INPUT #9,fc(i)                           [C650]
7160 NEXT i                                   [AE6C]
7170 INPUT #9,lr                              [B30A]
7180 FOR i=1 TO lr-1                          [BFDA]
7190 FOR j=1 TO af                            [10F4]
7200 INPUT #9,f$(i,j)                         [43F6]
7210 NEXT j                                   [D766]
7220 NEXT i                                   [E766]
7230 CLOSEIN                                  [7EF2]
7250 REM *** Nummerische Werte ablegen *
     **                                       [D886]
7260 FOR i=1 TO lr-1                          [DDD8]
7270 FOR j=1 TO af                            [A6F2]
7280 IF UPPER$(ft$(j))="N" THEN 7290 ELS
     E 7300                                   [1D3A]
7290 f$(i,j)=SPACE$(fc(j)-LEN(f$(i,j)))+
     f$(i,j)                                  [9E54]
7300 NEXT                                     [F452]
7310 NEXT                                     [1854]
7320 hvar1=0:FOR i=1 TO af:hvar1=hvar1+f
     c(i):NEXT                                [25BE]
7330 hvar2=FIX((80-hvar1)/af)                 [3606]
7340 IF hvar2<1 THEN hvar2=1 ELSE IF hva
     r2>10 THEN hvar2=10                      [EA6E]
7350 RETURN                                   [8E9E]
8000 REM *** Packen der Datei ***             [B6B8]
8010 FOR i= sn TO lr-1                        [976C]
8020 FOR j=1 TO af                            [A6E6]
8030 f$(i,j)=f$(i+1,j)                        [2170]
8050 NEXT j                                   [6D6C]
8060 NEXT i                                   [156C]
8065 lr=lr-1                                  [0094]
8070 RETURN                                   [079E]
10000 REM *** PROGRAMM ENDE ***               [BAE4]
10010 CLS                                     [BFE8]
10020 GOSUB 200:GOSUB 220                     [A666]
10023 LOCATE #2,28,2: PRINT #2,"P R O G
      R A M M{3 SPACE}E N D E";               [A4BE]
10030 REM                                     [7CF0]
10040 LOCATE 1,8: INPUT "Haben Sie sich
      das gut ueberlegt ? ",a$                [451A]
10050 IF UPPER$(a$)<>"J" THEN RETURN          [B944]
10060 MODE 2: BORDER 0: END                   [B54E]
```

Listing. Datenverwaltung (Schluß)

Glatter Bruch



Einem echten Bruch sieht man nicht von vornherein an, wie er als Dezimalzahl ausschaut. Das Programm »Division« zeigt einen Bruch mit allen Einzelheiten.

Nach »RUN« erwartet das Programm die Eingabe von Zähler und Nenner. Jetzt werden Sie denken, daß Ihr CPC auch ohne dieses Programm Brüche berechnen kann, wenn Sie beispielsweise eingeben »PRINT 2/3«. Richtig. Aber was Ihr Computer als Ergebnis bringt, ist schlicht unbefriedigend. Erstens zeigt er nur neun Stellen nach dem Dezimalpunkt. Und dann verschweigt er Ihnen glatt wichtige Informationen. Hier setzt das Programm »Division« an. So erfahren Sie zum Beispiel, daß 93727 geteilt durch 11300 einem gemischt-periodischen, unendlichen Dezimalbruch entspricht, wobei die Periode an der dritten Nachkommastelle beginnt und jeweils 112 Ziffern lang ist. Auf dem Bildschirm wird außerdem das Ergebnis mit 114 Stellen hinter dem Komma dargestellt.

(Wolfgang Jäger/ja)

```
100 MODE 0:LOCATE 1,11:PRINT STRING$(19,
    "*")                                     [F15C]
110 PRINT:PRINT"* D I V I S I O N *"         [4346]
120 '                                        [E0B4]
130 '*** (C) - Okt./1985 - Wolfgang A. J
    AEGER           ***                      [DCB2]
140 '*** <Version 2.2>      PF 1731, 7600
     Offenburg 1 ***                         [50E8]
150 '                                        [E1BA]
160 PRINT:PRINT STRING$(19,"*")              [03F6]
170 LOCATE 3,25:PRINT"- bel. Taste -"        [70FA]
180 DEFINT i,n,y,r                           [CAB2]
185 DEFREAL x                                [B792]
190 FOR i=1 TO 1000:IF INKEY$="" THEN NE
    XT                                       [78F0]
200 MODE 2                                   [AB52]
210 LOCATE 1,10:INPUT"Zaehler = ",x          [4B86]
220 IF ABS(x)<>INT(x) OR x=0 THEN 210        [A5F0]
230 LOCATE 1,14:INPUT"Nenner{2 SPACE}= "
    ,y                                       [160A]
240 IF ABS(y)<>INT(y) OR y*y=y THEN 230      [E3D6]
250 FOR i=1 TO 500:IF INKEY$="" THEN NEX
    T                                        [2392]
260 CLS:PRINT:PRINT                          [8A50]
270 PRINT RIGHT$(STR$(x),LEN(STR$(x))-1)
    "/"RIGHT$(STR$(y),LEN(STR$(y))-1)"="
    RIGHT$(STR$(INT(x/y)),LEN(STR$(INT(x
    /y)))-1)".";                             [D178]
280 IF y/x>1 THEN n=ROUND(LOG10(y/x)+1)
    ELSE n=0                                 [CF7E]
290 n=y+n                                    [2AF0]
300 DIM rest(n)                              [3A54]
310 FOR i=1 TO n                             [6DC6]
320   x=x-y*INT(x/y):rest(x)=rest(x)+i       [0CCE]
330   IF x=0 THEN PRINT:GOTO 390             [C26A]
340   IF i=1 THEN 360                        [050A]
350   IF rest(x)<>i THEN PRINT:GOTO 400      [8368]
360   x=x*10                                 [FC62]
370   PRINT RIGHT$(STR$(INT(x/y)),1);        [BCF8]
380 NEXT i:PRINT:PRINT                       [3722]
390 LOCATE 1,20:PRINT"> Endliche Dezimal
    zahl":PRINT"{2 SPACE}mit"i-1"Nachkom
    mastellen":GOTO 470                      [CD82]
400 LOCATE 1,20:PRINT"> Unendliche Dezim
    alzahl "                                 [57D0]
410 IF rest(x)<>i+1 THEN 430                 [DB60]
420 PRINT"> Rein-periodisch{8 SPACE}":PR
    INT"> Periodenlaenge:"i-1" ":GOTO 47
    0                                        [66D0]
430 PRINT"> Gemischt-periodisch{4 SPACE}
    "                                        [4CE2]
440 PRINT"> Periodenbeginn an der "          [B5F4]
450 PRINT "{2 SPACE}"RIGHT$(STR$(rest(x)
    -i),1)"ten Nachkommastelle "             [0550]
460 PRINT"> Periodenlaenge:"2*i-rest(x)"
    "                                        [AF44]
470 LOCATE 1,25:PRINT"(1) neue Division{
    6 SPACE}(2) Ende "                       [982A]
480 a$=INKEY$:IF a$="" THEN 480              [E838]
490 IF a$="1" THEN RUN 180                   [3C80]
500 IF a$<>"2" THEN 480                      [91C8]
510 MODE 1::END                              [B4EE]
```

Listing. Jetzt kann der CPC »richtig« rechnen

Weltstandard mit einem 8080-Assembler

Die Schneider-Computer stammen, auch wenn das die Manager von Amstrad und Schneider nicht so gerne hören, aus der Heimcomputer-Ecke. Schließt man allerdings ein Diskettenlaufwerk an und benutzt das Betriebssystem CP/M der mitgelieferten System-Diskette, so wandelt sich das Gerät in einen Computer, der professionelle Programme bearbeiten kann. Die Software-Hersteller haben das erkannt und bieten Ihre Produkte zu Preisen an, die der normalen Heimcomputer-Software entspricht.

Das Thema CP/M kann man aber auch andersherum betrachten. CP/M-Programme auf dem Schneider lassen sich fast ohne Probleme auf andere CP/M-Computer übertragen. Das Hand-



Schreiben Sie die Software für Ihren Schneider unter CP/M! Dann läuft sie auch auf tausend anderen Computern, denn CP/M ist ein echter Weltstandard. Und so geht's!

buch geht leider auf das Betriebssytem fast überhaupt nicht ein. Dabei gibt es viele Bereiche, in denen man ohne CP/M nicht auskommt.

Die großen Softwarehäuser stehen schon immer vor einem nahezu unlösbaren Problem:

Sobald ein neuer Computer vorgestellt wird, sollen von Anfang an möglichst viele Programme lieferbar sein. Meist können die Firmen auch sehr viel Software liefern – doch leider nur für das Konkurrenzmodell. Jedes Programm muß so von Grund auf neu entwickelt werden, damit es an die jeweilige Hardware des neuen Gerätes optimal angepaßt ist. Während die Firmen eifrig Software für den neuen Computer schreiben, vergehen meist Monate, bis die ersten Programme fertig sind und verkauft werden können. In der Zwischenzeit kann viel passieren. Das Schlimmste von allem ist wohl, daß sich das Gerät als Flop erweist und der Hersteller die Produktion einstellt. Dann sind die ganzen Investitionen der Softwarehäuser »in den Sand gesetzt«. Da man solche Risiken nur schwer abschätzen kann, entsann man sich einer simplen,

aber doch äußerst wirksamen Lösung. Ein Standard-Betriebssystem war gefragt. Dieses mußte genau definierte Programmiermöglichkeiten bieten und leicht auf eine Vielzahl von Computern anzupassen sein. Da dann die Programme praktisch unverändert übernommen werden können, sparen die Firmen einen Großteil ihrer Entwicklungskosten und können innerhalb kürzester Zeit lauffähige Software für eine ganze Reihe verschiedener Computer auf den Markt bringen.

Am Anfang stand die Vielfalt

Es gibt eine Vielzahl solcher Standard-Betriebssysteme, beispielsweise MS-DOS, UNIX, FLEX, OS-9 und in gewisser Weise auch das von den Atari-Computern her bekannte GEM. Diese Betriebssysteme sind meist genau auf einen Mikroprozessor zugeschnitten. Da im 8-Bit-Bereich der Z80 sowie der 8080 und der 8085 eine dominierende Marktstellung besitzen, ist hier auch das CP/M-Betriebssystem, das mit diesen drei Prozessoren arbeitet, sehr weit verbreitet.

Das Betriebssystem CP/M wurde von der amerikanischen Firma Digital Research entwickelt. Diese Firma brachte eine ganze Familie von CP/M-Betriebssystemen auf den Markt, die in ihrer Fülle leicht verwirren können. Da schwirren Begriffe wie CP/M 1.0, CP/M 2.2, CP/M 3.0, CP/M Plus, CP/M-86, CP/M-80, CP/M-86/80, CCP/M, MP/M und einige andere durch den Raum.

Fangen wir der Reihe nach an. CP/M 1.0 ist die erste vorgestellte Version dieses Betriebssystems und wird heute von so gut wie niemandem mehr benutzt. CP/M 2.2 wurde gegenüber dieser Version stark erweitert und von Programmierfehlern befreit. Heute ist CP/M 2.2 die am weitesten verbreitete Version und befindet sich auch bei den Computern der CPC-Reihe im Lieferumfang. CP/M 3.0 und CP/M Plus sind ein und dasselbe: eine Fortentwicklung von CP/M 2.2. Die hervorstechendste Verbesserung ist die Fähigkeit, mehr als 64 KByte Speicher verwalten zu können. Es gibt aber auch eine »abgespeckte« Version von CP/M 3.0, die mit 64 KByte auskommt. Sie benötigt erheblich mehr Speicherplatz als CP/M 2.2. Ein Benutzer wird deshalb kaum auf sie zurückgreifen. CP/M-80 ist ein etwas mißverständlicher Name, denn er sagt nur aus, daß das Betriebssystem für Mikroprozessoren der 80er-Klasse, also den 8080 und 8085 des Chipherstellers Intel und den Z80 von Zilog geeignet ist. Darunter würden also alle Versionsnummern von 1.0 bis 3.0 fallen. Im allgemeinen Sprachgebrauch ist damit aber immer die Version 2.2 gemeint.

Die anderen Namen der Betriebssysteme brauchen Sie eigentlich gar nicht zu interessieren, da diese auf dem Schneider nicht implementiert werden können. CP/M-86 läuft auf den Prozessoren 8086, 8088, 80186 und 80286 (alle von Intel). Bei diesen CPUs handelt es sich um 16-Bit-Prozessoren, die vor allem beim IBM-PC und dessen Kompatiblen Verwendung finden. Somit steht CP/M-86 mit Microsoft MS/PC-DOS in Konkurrenz. CP/M-86/80 ist eine Spezialentwicklung von Digital Equipment für den Rainbow 100. Dieses CP/M verarbeitet gleichermaßen 8-Bit- und 16-Bit-Software. Die übrigen Systeme – CCP/M, MP/M und was es sonst noch gibt – sind Versionen, die das gleichzeitige Abarbeiten mehrerer Programme oder die Bedienung mehrerer Benutzer, die an verschiedenen Terminals des gleichen Computers sitzen, ermöglichen.

Natürlich müssen Sie nicht alle diese Versionen auseinanderhalten können. Merken Sie sich nur, daß allein CP/M-2.2 (CP/M-80) und CP/M-3.0 (CP/M-Plus) für Sie interessant sind. Wenn wir im folgenden von CP/M sprechen, ist immer die Version 2.2 gemeint, sofern nichts anderes erwähnt.

CP/M und Maschinensprache sind untrennbar miteinander verbunden

Zu CP/M gehört immer die Maschinensprache des 8080-Prozessors von Intel. Auch wenn im Schneider ein Z80 von Zilog eingebaut ist, ist dieses Standard-Betriebssystem in 8080-Assembler geschrieben. Der Grund dafür ist einfach. Der Befehlssatz des 8080 und auch des 8085 ist ein Teil des Z80-Befehlsvorrats. Oder andersherum: Der Z80-Prozessor ist eine Weiterentwicklung des 8080 und des 8085. Da CP/M möglichst universell einsetzbar sein sollte, hat sich Digital Research dazu entschlossen, auf die zusätzlichen Z80-Befehle zu verzichten und deshalb das Betriebssystem in 8080-Assembler entwickelt. Sofern Sie Software schreiben wollen, die auf allen CP/M-Computern lauffähig sein soll, können Sie somit die Besonderheiten des Z-80-Prozessors nicht ausnützen, da einige CP/M-Geräte immer noch die Intel-Prozessoren verwenden. Einem eingefleischten Z80-Programmierer wird es in der Seele leid tun, wenn er die leistungsfähigen Erweiterungen »seines« Prozessors nicht immer benutzen darf, zum Beispiel die relativen Sprünge, die Blockverschiebe- und Blocksuchbefehle, die beiden zusätzlichen Indexregister IX und IY, die Bit-Befehle SET, RES und BIT und die komfortablen Schiebe- und Rotierkommandos. Man kann aber davon ausgehen, daß etwa 95 Prozent der heutigen CP/M-Computer mit einer Z80-CPU ausgerüstet sind. Verwendet werden heute die Intel-Typen vorrangig in Hand-Held-Computern, weil es eine CMOS-Version 80C85 gibt, die besonders stromsparend arbeitet. Und darauf kommt es bei Batteriebetrieb eben an. Die Mengenverhältnisse der beiden Prozessorgruppen haben aber inzwischen auch schon einige Softwarefirmen erkannt und Programme auf den Markt gebracht, die ausschließlich mit einem Z80 laufen. Prominentestes Beispiel dafür dürfte Borland International mit Turbo-Pascal sein.

Wenn Sie sich nicht mit Maschinensprache beschäftigen wollen, aber dennoch planen, CP/M-Programme zu schreiben, können Sie auf eine Vielzahl von Compilersprachen zurückgreifen. Es gibt eine ganze Reihe von C-Compilern, ebenso PL/I, Cobol, Fortran und nicht zuletzt Turbo-Pascal. Allerdings müssen Sie sich diese Compiler erst kaufen, während ein Assembler (beim CPC 6128 sind es sogar deren drei) zum Lieferumfang der Diskettenstation gehört. Auch werden Sie in einer Hochsprache nie die volle Leistungsfähigkeit eines gleichartigen Maschinenprogramms erreichen, schon allein deshalb, weil der Aufruf von Betriebssystem-Routinen in den meisten Compilersprachen einige Probleme hervorruft. Deshalb geben wir Ihnen im folgenden eine Einführung in die 8080-Maschinensprache.

Wollen Sie die Einführung in Assembler verstehen, ist es sehr hilfreich, wenn Sie in Basic einige Grundlagen und den Unterschied zwischen Dezimal-, Hexadezimal- und Binärzahlen kennen. Haben Sie damit noch Probleme, so ist es besser, wenn Sie erst Ihre Kenntnisse in Basic vertiefen, bevor Sie auf CP/M umsteigen. Das zum Lieferumfang der Schneider-Computer gehörige Handbuch enthält übrigens eine Einführung in Binär- und Hex-Arithmetik.

Im Locomotive-Basic, das sich beim Schneider im ROM befindet, werden Hexadezimalzahlen durch ein vorangehendes kaufmännisches Und-Zeichen (&) dargestellt. Verschiedene Assembler, beispielsweise der Devpac-Assembler von HiSoft ziehen das Doppelkreuz (#) vor, während der CP/M-Assembler ein nachgestelltes H verlangt. Also ist zum Beispiel 2000H gleich &2000 oder #2000.



Die 80er-CPUs sind 8-Bit-Prozessoren. Das heißt, ihr Datenbus ist 8 Bit breit. Der Datenbus sind die Beinchen am Chip, über die der Prozessor Daten mit dem Speicher austauscht. Der zweite Bus ist der Adreßbus. Dieser hat eine Breite von 16 Bit. Damit lassen sich also 2^{16} Byte (= 65536 Byte = 64 KByte) adressieren. Immer wenn der Chip in den Speicher ein Byte schreiben oder aus diesem ein Byte lesen will, muß er den Speicherbausteinen über den Adreßbus die gewünschte Adresse mitteilen, und kann sich dann das Byte über den Datenbus abholen, beziehungsweise über diesen Bus an die RAM-Chips schicken. Über den Datenbus holt sich der Prozessor sowohl seine Programmierbefehle als auch die zu verarbeitenden Daten. Der Mikroprozessor tut also nichts anderes, als sich dauernd aus dem Speicher Befehle und die zugehörigen Daten zu besorgen, diese zu kombinieren und wieder zurückzuschreiben. Dies macht er solange, bis ihm der Strom abgeschaltet wird.

Jeder Mikroprozessor arbeitet mit Registern. Register sind Speicherplätze im Prozessor, sozusagen ein Mini-RAM in der CPU. In diesen Registern kann der Programmierer die wichtigsten, also regelmäßig benötigten Daten aufbewahren. Register lassen sicht entfernt mit den Variablen in Basic vergleichen. Man kann mit diesen Registern rechnen und sie miteinander beispielsweise logisch verknüpfen. Es ist klar, daß ein Mikroprozessor um so einfacher zu programmieren ist, je mehr Register er besitzt. Die 8080-CPU ist für den 8-Bit-Bereich ein in dieser Hinsicht sehr komfortabler Prozessor, denn es gibt sieben frei verwendbare Register. Diese erhalten als Bezeichnung je einen Buchstaben. A,B,C,D,E,H und L sind die Registernamen. Die Register können genau so viele Bits aufnehmen, wie auf den Datenbus passen: 8 Bit. Damit lassen sich 2^8, also 256 Zahlen darstellen.

8080-Assembler kommt zu ungeahnten Ehren

Die Inhalte der Register lassen sich beliebig austauschen. Dazu dient der MOV-Befehl. MOV steht für »Move« und heißt übersetzt »Bewege« oder – freier – »kopiere nach«. Ein kleines Beispiel: MOV D,H heißt, daß der Inhalt des H-Registers ins D-Register geladen wird. Dabei bleibt das H-Register natürlich unverändert, denn der Prozessor legt – wie gesagt – nur eine Kopie des Werts im anderen Register an. Achten Sie aber auf die Reihenfolge, in der die Operanden (die Register) angegeben werden.

Weitere Beispiele für den MOV-Befehl sind MOV A,C; MOV H,L; MOV L,H; MOV D,C und so weiter. Wenn Sie in Basic Variablen mit den gleichen Namen wie die Register verwenden, können Sie die Beispiele dort so schreiben:

»A=C«,»H=L«,»L=H«,»D=C«.

Nun hätte es wenig Sinn, ständig die Inhalte der Register hin- und herzukopieren, ohne eine Möglichkeit zu

Rechnen kann er auch

haben, ein Register direkt mit einer Zahl zu laden. Dazu dient der Befehl MVI (Move Immediate, »Lade den unmittelbar folgenden Wert«). Geben Sie zum Beispiel den Befehl MVI C,26, dann erhält das C-Register den Wert 26 zugewiesen. In Basic hieße das »C=26«. Der größte Wert, den ein solches Register speichern kann, ist 255, der kleinste die Null. Es sind nur ganze Zahlen erlaubt, keine Fließkommawerte wie 3,14 oder 0,27.

Bisher wurde stillschweigend vorausgesetzt, daß alle Register gleichwertig sind. Dem ist aber nicht so, denn das A-Register hat viel mehr zu tun als alle übrigen Register. Das A-Register wird auch Akkumulator oder kurz Akku genannt. Wenn Sie sich die lateinische Bedeutung des Wortes anschauen, bekommen Sie vielleicht eine Ahnung von den Aufgaben dieses Registers: »accumulare« heißt »anhäufen« oder »ansammeln«. Dieses Ansammeln ist durchaus wörtlich zu nehmen. Alle 8-Bit-Rechenbefehle benötigen den Akku als den einen der beiden Operanden und schreiben das Ergebnis wieder in den Akkumulator. So »sammelt« sich mit der Zeit im Akkumulator das Rechenergebnis an.

Die »grundlegendste« aller Grundrechenarten ist die Addition. Dazu gibt es folgende Addierbefehle:

ADD A, ADD B, ADD C, ADD D, ADD E, ADD H und ADD L.

Da der Akku zu diesen Befehlen immer benötigt wird, wird er gar nicht mehr angegeben. ADD B addiert also den Inhalt des B-Registers zum Inhalt des Akkus und speichert das Ergebnis wieder im Akku.

Versuchen Sie noch einmal, anhand der bereits bekannten Befehle das nachfolgende Miniprogramm zu analysieren!

```
1     MVI A,3
2     MVI C,45
3     ADD C
```

Zuerst lädt das Programm den Akku mit dem Wert 3 und das C-Register mit der Zahl 45. In Zeile 3 addiert der Computer den Inhalt des C-Registers (45) zum Inhalt des Akkus (3) und speichert das Ergebnis wieder im Akku ab. Dort steht also 48. Das C-Register bleibt natürlich unverändert und behält den Wert 45.

Merken Sie sich also: Bei den 8-Bit-Arithmetikbefehlen ist immer der Akku einer der beiden Operanden und das Ergebnis wird immer im Akku abgelegt.

Er gibt auch einen Befehl, der zum Akkuinhalt direkt – ohne Umweg über ein Register – einen Wert addiert. ADI steht immer für »Add Immediate«, (Addiere den unmittelbar folgenden Wert). Das obige 3-Zeilen-Programm läßt sich mit diesem Befehl vereinfachen:

```
1      MVI A,3
2      ADI 45
```

Der Unterschied liegt darin, daß das C-Register bei diesem Programm nicht benötigt wird. Es kann also für andere Zwecke freigehalten werden.

Wenn es Additionsbefehle gibt, muß es auch Befehle zur Subtraktion zweier Zahlen geben. Analog zu ADD und ADI sind sie mit SUB und SUI benannt. Subtraktionen sind wieder mit allen Registern möglich, wobei der Akku den Wert enthält, von dem der andere Wert abgezogen wird:

```
1     MVI A,26
2     MVI C,10
3     SUB C
```

Der Comptuer führt die Rechnung 26 – 10 = 16 durch und speichert das Ergebnis im Akku. Auch dieses Programm läßt sich mit dem Immediate-Befehl SUB (Subtract Immediate) vereinfachen:

```
1     MVI A,26
2     SUI 10
```

Diese zwei Befehle erfüllen dieselbe Aufgabe wie die vorherigen drei, belegen aber wiederum kein zusätzliches Register.

Zählen ist erheblich einfacher

Da in Maschinencode-Programmen sehr häufig der Inhalt der Register um Eins erhöht oder vermindert werden muß, wurden Befehle in den Chip integriert, mit denen man jedes Register direkt – also nicht über den Akkumulator – dekrementieren (um Eins vermindern) oder inkrementieren (um Eins erhöhen) kann. Diese Befehle werden schneller als die Addition ausgeführt und brauchen nur ein Byte Speicher (im Gegensatz zu den zwei Byte bei der Addition). Sie heißen DCR (Decrement) und INR (Increment):



DCR A, DCR B, DCR C, DCR D, DCR E, DCR H, DCR L sowie INR A, INR B, INR C, INR D, INR E, INR H und INR L.

Ein einfaches Beispiel für beide Befehle soll hier genügen:

```
1    MVI  C,26
2    DCR  C          C ist jetzt 25
3    MVI  E,24
4    INR  E          E ist jetzt
                       auch 25
```

Wenn Sie sich jetzt fragen, wann die Multiplikations- und Divisionsbefehle erklärt werden, müssen wir Sie enttäuschen. Der 8080-Prozessor kennt – wie die Mehrzahl der 8-Bit-Prozessoren – keine Punktrechnung. Diese Berechnungen müssen mit Hilfe von selbstprogrammierten Routinen durchgeführt werden. Schließlich läßt sich beispielsweise eine Multiplikation – wenn auch umständlich – durch eine Reihe von Additionen ersetzen.

Logeleien im 8080

Statt dieser beiden Rechenarten beherrscht unsere CPU vorzüglich eine Reihe von logischen Befehlen, nämlich UND, ODER, EXCLUSIV-ODER und NICHT. Diese Logikbefehle arbeiten auf Bit-Ebene und verknüpfen der Reihe nach jeweils die beiden Bits zweier Zahlen, die an der gleichen Stelle stehen, also die gleiche Wertigkeit besitzen. Was hier recht kompliziert klingt, läßt sich einfach im Binärsystem darstellen. Dazu dienen uns zwei Binärzahlen als Beispiel:

0A hex = 10 dezimal = 0000 1010
A6 hex = 16 dezimal = 1010 0110

Beim logischen Oder wird im Ergebnis ein Bit gesetzt, wenn in einer der beiden Zahlen – oder auch in beiden – jeweilige Bit gesetzt sind:

```
        0000  1010
oder    1010  0110
--------------------
        1010  1110
```

Es gilt hier: 0 or 1=1, 1 or 0=1, 0 or 0=0 und 1 or 1=1.

Sie können Rechnungen auch in Basic durchführen. Dort gibt es alle 8080/Z80-Logikbefehle auch. Sie heißen dort: AND, OR, XOR und NOT. Tippen Sie also unter Basic ein: »PRINT BIN$ (&X00001010 OR &X10100110«, und Sie erhalten das Ergebnis »10101110«.

Wie bei den Arithmetikbefehlen gibt es zwei ODER-Kommandos. Eines zur Verknüpfung des Akkus mit einem Register, nämlich ORA (Or Accumulator, ODER Akku) und ORI (Or Immediate, Akku ODER unmittelbar folgender Wert) zur Verknüpfung des Akkus mit einem direkt angegebenen Wert. Einer der Operanden muß also immer im Akku stehen. Die obige Rechnung läßt sich in Maschinencode und Basic so umsetzen:

```
1    MVI A,0AH - A=&0A
2    MVI D,A6H - D0&A6
3    ORA D - A=A OR D
```

Da der Akku beim ORA-Befehl mit allen Registern geODERt werden darf, sind die Befehle ORA A, ORA B, ORA C, ORA D, ORA E, ORA H und ORA L zulässig.

Der ORI-Befehl, der die unmittelbare Angabe einer Zahl gestattet, verkürzt das Programm auf zwei Zeilen:

```
1 MVI A,0AH - A=&0A
2 ORI A6H - A=A OR &A6
```

Das logische UND ist etwas anders definiert. Ein Bit wird gesetzt, wenn sowohl in der ersten als auch in der zweiten Zahl dieses Bit gesetzt ist. Andernfalls wird es gelöscht (»zurückgesetzt«):

```
        0000 1010
und     1010 0110
-------------------
        0000 0010
```

Diese Regeln lauten damit beim logischen UND: 0 and 1=0, 1 and 0=0, 0 and 0=0, 1 and 1=1.

Die zwei Befehle heißen ANA (And Accu, Akku UND Register) und ANI (And Immediate, Akku UND folgende Zahl). In Assembler und Schneider-Basic schaut die Rechnung so aus:

```
1 MVI A,0AH - A=&0A
2 MVI E,A6H - E=&A6
3 ANA E - A=A AND E
```

Wenn Sie sich fragen, warum einmal das C-Register, später das D-Register und dann das E-Register zur Speicherung des Wertes dienen: Abwechslung muß sein! Sie können genauso gut ein beliebiges anderes Register verwenden. Denn die erlaubten Befehle sind ANA A, ANA B, ANA C, ANA E, ANA H und ANA L. Einfacher wird es wieder mit dem ANI-Befehl:

```
1    MVI A,0AH - A=&0A
2    ANI A6H - A=A AND &A6
```

Negativ oder positiv

Der Exklusiv-Oder-Befehl ist eine Abwandlung des ODER-Befehls. Hier wird ein Bit nur gesetzt, wenn eines der beiden Bits gesetzt ist, aber nicht beide. Die Befehle lauten analog zu den anderen Logik-Befehlen XRA (EXCLUSIVE OR ACCU) und XRI (EXCLUSIVE OR IMMEDIATE). Es gelten die folgenden Regeln: 0 xor 1=1, 1 xor 0=1, 1 xor 1=0, 0 xor 0=0. Versuchen Sie doch selbst einmal, ein Basic-Programm und ein passendes Maschinenprogramm zu schreiben.

Der folgende CMA-Befehl fällt von der Syntax her etwas aus der Reihe. Er braucht nur einen Operanden. Dieser steht im Akku, wo auch das Ergebnis abgelegt wird. CMA steht für Complement Accumulator und bildet das Einer-Komplement des Akkuinhalts. Was sich jetzt sehr großartig anhört, bedeutet ganz einfach, daß alle Bit im Akku herumgedreht werden. Aus Einsen werden Nullen und aus Nullen werden Einsen:

```
1    MVI A,1 - A=1
2    CMA - A=NOT A
```

Die Dezimale 1 sieht binär so aus: 0000 0001. Bei der Umwandlung werden alle Bits umgewandelt: 1111 1110. Heraus kommt die Zahl 254. Wenn Sie den Befehl »PRINT NOT 1« in Basic ausprobieren, werden Sie aber nicht 254, sondern -2 als Ergebnis erhalten.

Bisher haben wir vorausgesetzt, daß ein Register nur Zahlen zwischen Null und 255 speichern kann. Um auch negative Zahlen darstellen zu können, hat man vereinbart, daß man das höchstwertige Bit einer Zahl, also das erste Bit von links, als Vorzeichenbit betrachten kann.

Ist dieses Bit auf Null gesetzt, handelt es sich um eine positive Zahl, andernfalls ist der Wert negativ. Da dieses Bit aber nicht mehr zur Darstellung der Zahl zur Verfügung steht, bleiben nur noch 7 Bit für die Zahl frei. Mit diesen lassen sich nur die Werte von 0 bis 127 darstellen. Das Vorzeichenbit erlaubt dann noch die Auswahl zwischen positiven und negativen Zahlen. Somit reicht der Zahlenbereich bei vorzeichenbehafteter Darstellung von -128 bis +127. Dazu ein Beispiel:

Die Zahl A0 hex (160) sieht binär so aus: 1010 0000. Nimmt man das erste Bit von der Zahl weg und betrachtet man es als Vorzeichen, so stellt man fest, daß das Bit gleich Eins und die Zahl damit – für den Computer – negativ ist. Für die Zahl bleiben noch die Bits 010 0000 übrig, 32 dez. Also ist 160 für den Computer gleich -128+32=-96.

Zurück zum Beispiel: Die Zahl 254 lautet in Binärdarstellung »1111 1110«. Folglich ist sie negativ und der Zahlenwert ist »1111 1110«, 126 dez. -128 +126 ist aber -2 und wir wissen, woher die negative Darstellung unseres Basic-Befehls kommt. Denken Sie aber immer daran, daß die vorzeichenbehaftete und die vorzeichenlose Darstellung gleichzeitig nebeneinander verwendet werden dürfen.

Sie werden nun fragen, wozu man um alles in der Welt diese umständlichen Logik-Befehle benötigt. Als Antwort soll vorerst genügen: Sie werden diese Befehle noch sehr viel häufiger brauchen, als Sie es jetzt vielleicht erwarten. Ein kleines Beispiel, das die Nützlichkeit dieser Befehlsgruppe zeigt. Im Akku steht eine beliebige Zahl zwischen 0 und 255. Dieser Wert soll einem Unterprogramm übergeben werden, darf aber nur im Bereich zwischen 0 und 63 liegen:

```
1 MVI A,?? - A=?? (beliebiger
  Wert)
2 ANI 63 -A=A AND 63
```

Diese Methode funktioniert aber nur, wenn der Grenzwert, hier 63, eine Zahl ist, die sich aus einer Zweier-Potenz minus Eins herleiten läßt, also: 1, 3, 7, 15, 31, 63, 127 und 255. Aber auch anders herum ergibt die Sache Sinn. Bei einer Zahl soll sichergestellt werden, daß das höchstwertige Bit (das erste Bit von links) auf Eins liegt:

```
1 MVI A,?? - A=?? (beliebig)
2 ORI 128 - A=A OR 128
```

Vor einem Sprung die Fahne hoch

Es ist natürlich wenig sinnvoll, ein Programm zu schreiben, das vom ersten bis zum letzten Befehl linear durchgearbeitet wird. Vielmehr enthalten fast alle Programme Vergleiche und abhängig von deren Ergebnis Sprungbefehle. Ähnlich dem IF-Befehl in Basic gibt es hier ein mnemonisches Kommando mit dem Namen CMP für Compare (auf Deutsch »Vergleiche«). Dieser Befehl vergleicht den Inhalt des Akkus mit dem des angegebenen Registers, beispielsweise CPM A,C. Auch ein Direktbefehl ist vorhanden: CPI (Compare Immediate). Dieser vergleicht den Akkuinhalt mit der dem Befehl folgenden Zahl, zum Beispiel CPI 200.

Der Computer muß sich das Resultat des Vergleichs irgendwo merken können. Dazu benutzt er ein Register, das bisher noch nicht besprochen wurde und auch völlig andere Aufgaben als die besprochenen Register hat: das Flag-Register oder kurz F-Register. Flag heißt auf Deutsch »Flagge«. Dieses Register ist bitweise organisiert und besteht aus acht Einzelbits. Für jeden Prozessorzustand ist ein Bit reserviert – sozusagen als »Flagge«, die gehißt wird, wenn ein bestimmter Zustand eingetreten ist. Da der Computer abhängig vom Zustand eines Flagbits eine Entscheidung treffen kann, gibt es Sprungbefehle zu verschiedenen Adressen: Der unbedingte Sprungbefehl JMP (Jump heißt Springen). Er ist in etwa vergleichbar mit GOTO in Basic und zwingt den Computer, die Abarbeitung des Programms an einer anderen Stelle im Speicher fortzusetzen. Dieser JMP-Befehl ist also an keine Bedingungen geknüpft. Es gibt aber eine Reihe von bedingten Sprungbefehlen, die jeweils mit einem ganz bestimmten Flagbit korrespondieren.

Die folgenden Bits des F-Registers sind auch für den Programmierer von Bedeutung:

C = Carry
Z = Zero Flag
M = Minus Flag
E = Even Parity Flag
I = Interdigit Flag

Das Carry-Flag wird gesetzt, wenn bei Additionen oder Subtraktionen ein Übertrag auftritt. Es wird auch gesetzt, wenn bei Vergleichen mit CMP oder CPI der Akkuinhalt kleiner als der Wert ist, mit dem er verglichen wird. Zurückgesetzt wird das Carry-Flag, wenn der Akkuinhalt größer oder gleich dem anderen Wert ist oder wenn bei Addition beziehungsweise Subtraktion kein Übertrag auftritt. Die zugehörigen Sprungbefehle heißen JC xxxx (Jump If Carry Flag Set, »Sprung, wenn das Carry-Flag gesetzt ist«) und JNC xxx (Jump if No Carry, »Sprung, wenn das Carry-Flag nicht gesetzt ist«). Damit lassen sich sehr leistungsfähige Programmstrukturen entwickeln, ein einfaches Beispiel dafür ist:

```
1              MVI A,10
1              CPI ??
3              JC KLEINER
4 GROESSER:    ...
5 KLEINER:     ...
```

In diesem Programm wird zuerst der Akku mit dem Wert 10 geladen. In Zeile 2 führt der Computer einen Vergleich mit einer – noch einzusetzenden – Zahl durch. Die Zeile 3 enthält einen Sprung zur Marke KLEINER, wenn das Carry-Flag gesetzt ist. KLEINER wird also aufgerufen, wenn der Akkuinhalt kleiner ist als der Wert, mit dem er verglichen wird. Andernfalls beachtet der Computer den Sprungbefehl nicht und macht bei der Marke GROESSER weiter.

Setzen wir doch einmal in den Vergleich (Zeile 2) konkrete Werte ein: CPI 11. Da 10 kleiner als 11 ist, setzt der Computer das Carry-Flag und arbeitet an der Marke KLEINER weiter. Ein anderes Beispiel: CPI 2. 10 ist größer als 2, also ist dem Mikroprozessor der Sprungbefehl in Zeile 3 egal, er hat ja das Carry-Flag zurückgesetzt.

Vergleichs-Weise: Flag oder nicht

Das Zero-Flag ist gesetzt, wenn zwei Werte bei einem Vergleich identisch sind. Die beiden Sprungbefehle lauten (entsprechend dem Carry-Flag) JZ xxxx (Jump If Zero Flag Set, »Sprung, wenn das Zero-Flag gesetzt ist«) und JNZ xxxx (Jump If Not Zero, »Sprung, wenn das Zero-Flag gelöscht ist«):

```
1              MVI A,3
2              CPI 3
3              JZ GLEICH
4 UNGLEICH:    ...
5 GLEICH:      ...
```

Hier wird der Akkuinhalt 3 mit der Zahl 3 verglichen. Da beide Werte natürlich identisch sind, setzt der Computer das Zero-Flag und springt zur Marke GLEICH. Setzen Sie nun einmal in Zeile 2 einen anderen Wert nach CPI ein und überlegen Sie die Resultate.

Das Minus-Flag ist auf Eins gesetzt, wenn der Inhalt des Akkumulators negativ ist. Man kann ja entweder 8-Bit-vorzeichenlos oder 7-Bit-vorzeichenbehaftet rechnen. Der Inhalt des Akkumulators ist negativ, wenn er zwischen –128 und –1 (7-Bit-Darstellung) beziehungsweise 128 und 255 (8-Bit-Darstellung) liegt.

GOTO gibt's auch in Maschinensprache

Die beiden Sprungbefehle, die sich ihre Informationen vom Minus-Flag holen, heißen JM (Jump If Minus, »Springe, wenn negativ«) und JP (Jump If Positive, »Springe, falls positiv«).

```
1              MVI  A,28
2              SUI  4
3              JM   NEGATIV
4              JP   POSITIV
5              MVI  A,28
6              SUI  32
7              JM   NEGATIV
8              JP   POSITIV
```

In den Zeilen 1 bis 4 ist das Ergebnis 24, also positiv. Demnach läßt der Computer den JM-Befehl unberücksichtigt und führt stattdessen den JP-Befehl aus. Anders sieht es in den Zeilen 5 bis 8 aus. Dort ist das Resultat -4, in 8-Bit-Darstellung $-4+256=252$, folglich negativ. Hier beachtet der 8080 bereits den JM-Befehl und arbeitet an der Marke NEGATIV weiter.

Das Even-Paritiy-Flag zeigt an, ob die Anzahl der gesetzten Bits im Akkumulator gerade oder ungerade (even oder odd) ist. Die Sprungbefehle lauten JPE (Jump If Parity Even, »Sprung, bei gerader Parität«) und JPO (Jump If Parity Odd, »Sprung bei ungerader Parität«). Dieses Flag hat allerdings keine allzu große Bedeutung.

Das Interdigit Flag zeigt an, ob ein Übertrag zwischen dem dritten und vierten Bit des Akkus aufgetreten ist. Dieses Flag läßt sich nur durch Tricks abfragen und hat für den Programmierer praktisch keine Bedeutung.

Der unbedingte Sprung, etwa vergleichbar mit GOTO in Basic, weist den Computer an, mit der Programmabarbeitung an der angegebenen Adresse fortzufahren. So löst ein JMP 0 beim Schneider einen Reset (Zurücksetzen des Computers in den Einschaltzustand) aus:

```
1    JMP 0
```

Fassen wir nochmals alle Jump-Befehle zusammen: JMP, JC, JNC, JZ, JNZ, JP, JM, JPE und JPO.

Sollen Unterprogramme aufgerufen werden, muß der Computer sich die Rücksprungadresse merken. Dazu dient der Befehl CALL. CALL xx ruft (unbedingt) ein Unterprogramm an der Adresse xx auf. Analog zu den Varianten des Jump-Befehls gibt es auch den bedingten Aufruf von Unterprogrammen:

CC xx – Call if Carry; Aufruf des Unterprogramms, wenn das Carry-Flag gesetzt ist.

CNC xx – Call If No Carry; Aufruf des Unterprogramms, wenn das Carry gelöscht ist.

CZ xx – Call If Zero; Aufruf des Unterprogramms, wenn das Zero-Flag gesetzt ist.

CNZ xx – Call If Not Zero; Aufruf des Unterprogramms, wenn das Z-Flag gelöscht ist.

CP xx – Call If Positive; Aufruf des Unterprogramms, wenn das Minus-Flag gelöscht ist.

CM xx – Call If Negative; Aufruf des Unterprogramms, wenn das Minus-Flag gesetzt ist.

CPE xx – Call If Parity Even; Aufruf des Unterprogramms bei gerader Parität.

CPO xx – Call If Parity Odd; Aufruf des Unterprogramms bei ungerader Parität.

Die Programmierung erfolgt genau wie bei den Jump-Befehlen:

```
1        MVI  A,3
2        CPI  26
3        CC   UNTERPROG
4        DCR  A
5        ADI  A,128
6        CM   UNTERPROG2
```

Damit der Computer wieder an die Aufrufstelle des Unterprogramms zurückkehren kann, verwenden Basic-Programmierer den RETURN-Befehl. So etwas gibt es auch im 8080-Assembler. RET holt die Rücksprungadresse und kehrt ins Hauptprogramm zurück. Auch hier lassen sich Bedingungen angeben: RC und RNC beziehen sich auf das Carry-Flag, während RM und RP vom Minus-Flag beeinflußt werden, und RPE und RPO bei gesetztem oder gelöschtem Parity-Flag einen Rücksprung auslösen. Ein typischer Unterprogramm-Aufbau könnte etwa so aussehen:

```
1        CALL UNTER
2        INR  A
3        JMP  0
4 UNTER: MVI  A,200
5        RET
```

Zuerst ruft der Computer das Teilprogramm UNTER auf. Dort lädt ein Maschinenbefehl den Akku mit der Zahl 200. RET weist den Prozessor an, an die Adresse nach dem Unterprogrammaufruf zu springen. Dort trifft er auf einen INR-Befehl, durch den der Akkuinhalt auf den Wert 201 gebracht wird. JMP 0 löst einen Reset des Computers aus. Wenn das Programm unter Amsdos, dem Basic-Betriebssystem, laufen soll, müßte dort ein RET-Befehl stehen, um ins Basic zurückzukehren. Denn der Computer betrachtet die ganze Maschinenroutine ebenfalls als Unterprogramm – als Unterprogramm des übergeordneten Betriebssystems. Sie sehen also, die RET-Befehle lassen sich beliebig verschachteln.

Es gibt noch eine Gruppe von acht speziellen Sprungbefehlen, die sogenannten Restarts (RST). Diese wirken ganz genauso wie der Call-Befehl, rufen aber nur acht genau festgelegte Adressen am Speicheranfang auf – nämlich 00 hex, 08 hex, 10 hex, 18 hex, 20 hex, 28 hex, 30 hex und 38 hex. Sie haben den Vorteil, statt drei Byte (wie beim Call-Befehl) nur ein Byte als Speicherplatz zu benötigen. Somit reservieren die meisten Systemprogrammierer die Restarts für die wichtigsten Unterprogramme eines Betriebssystems und sparen eine Menge der kostbaren Bytes. RST 0 löst beispielsweise einen Reset aus, die übrigen Restarts werden im Schneider zur Vewaltung der sich überlappenden RAM- und ROM-Bereiche benötigt. Der Rücksprung aus einem mit RST aufgerufenen Programm erfolgt übrigens genauso wie bei einem Call-Aufruf, also mit RET oder einem bedingten Return-Befehl.

Auch GOSUB und RETURN haben ihre Entsprechung

Der 8080-Prozessor ist tatsächlich ein halber 16-Bit-Prozessor, wenn er auch zu den 8-Bit-Chips zählt. Bisher wurden nur 8-Bit-Register angesprochen, nämlich der Akku und das Flag-Register sowie B, C, D, E, H und L. Als Besonderheit erlaubt es aber die CPU, zwei 8-Bit-Register zu einem 16-Bit-Register zusammenzufassen. So wird aus den Registern B und C das Doppelregister BC, aus D und E das Register DE und aus H und L das Register HL. Es gibt eine Reihe spezieller Befehle, die die Arbeit mit diesen 16-Bit-Registern ungemein erleichtern. Diese ähneln verschiedenen 8-Bit-Befehlen, haben aber andere Namen.

Haben Sie zum Beispiel bisher programmiert

```
1 MVI B,03H
2 MVI C,06H
```

läßt sich das mit dem LXI-Befehl vereinfachen. LXI heißt »Load Extended Register Immediate« oder »Lade Doppelregister unmittelbar«.

```
1 LXI B,0306H
```

Hier wird statt des 8-Bit-Registers B jetzt das 16-Bit-Register BD angesprochen. So können Sie Zahlen von Null bis 65535 (oder -32768 bis +32767, je nach Betrachtungsweise) auf einmal in ein Doppelregister laden und die 8-Bit- und 16-Bit-Befehle in Ihren Programmen beliebig mischen:

```
1  LXI D,0306H
2  DCR E
```

Nach diesen beiden Befehlen enthält das D-Register den Wert 03 hex, das E-Register aber 06 hex–1=05 hex. Das Doppelregister DE hat dann die Zahl 0305 hex gespeichert.

Das B-Register wird bei den 16-Bit-Befehlen als B, das DE-Register als D und das HL-Register als H bezeichnet. Es lassen sich immer nur die Register BC, DE und HL zusammenschalten. Andere Kombinationen wie etwa das B- mit dem L-Register sind nicht erlaubt.

Ein halber 16-Bit-Prozessor

Beachten Sie bitte, daß es keine Möglichkeit gibt, den Akkumulator auf 16-Bit-Breite zu erweitern. Stattdessen wird das HL-Register als eine Art 16-Bit-Akku betrachtet.

Es besteht die Möglichkeit, das HL-Register mit einer Adresse zu laden und dann Lade- oder Rechenbefehle zusammen mit dem Akku auszuführen. Dazu wird ein fiktives Register M verwendet, was ganz einfach »Memory« bedeutet. Ein Beispiel für eine Addition, bei der zum Akku der Inhalt der Adresse 2000 hex hinzugezählt wird:

```
1    MVI  A,45H
2    LXI  H,2000H
3    ADD  M
```

Das M-Register läßt sich also als die Adresse definieren, auf die das HL-Register zeigt. Steht zum Beispiel in 3000 hex der Wert FF hex und HL enthält den Wert 3000 hex, so addiert ADD M den Wert FF hex zum Akkuinhalt hinzu.

Hier noch ein besonderer Hinweis für alle, die sich ein wenig mit dem Z80-Prozessor auskennen: M wird dort nicht verwendet. Bei diesem Chip heißt es »HL«.

Viele der bisher besprochenen Befehle lassen sich so mit dem Pseudo-Register M verwenden: ADD M, ANA M, CMP M, DCR M, INR M, MOV A,M, MOV B,M, MOV C,M, MOV D,M, MOV E,M, MOV H,M, MOV L,M, MOV M,A, MOV M,B, MOV M,C, MOV M,D, MOV M,E, MOV M,H, MOV M,L, MVI M,xx, ORA M, SUB M und XRA M. Nicht erlaubt ist es hingegen, beim MOV-Befehl sowohl als Quelle als auch als Ziel anzugeben, zum Beispiel MOV M,M. Dazu müßte nämlich der 8080-Prozessor einen doppelten Adreßbus haben.

Bisher sind wir auf eine sehr nützliche Einrichtung des 8080-Prozessors noch nicht zu sprechen gekommen: den Stack. Sie haben ihn aber schon verwendet, ohne es zu wissen. Denn die Call- und Restart-Befehle legen dort die Rücksprungadresse ab.

(Hoch-)Stapelei ist nützlich

Was ist der Stack? Übersetzt heißt er »Stapelspeicher« und das beschreibt seine Aufgabe auch ziemlich genau. Er stapelt Zahlen aufeinander. Wenn Sie mehr Werte verwalten müssen, als Register frei sind, können Sie einen Registerinhalt mit einem einzigen Befehl dort ablegen und von dort wieder zurückholen.

Ein spezielles Register mit dem Namen SP (Stack Pointer) zeigt immer auf die letzte Eintragsadresse. Stellen Sie sich einen Tellerstapel vor: Sie legen immer wieder von oben einen Teller auf und nehmen von Zeit zu Zeit wieder einen Teller herunter. Dementsprechend wächst oder schrumpft der Stapel. Wenn Sie neben den Tellerstapel einen Zollstock oder ein Metermaß stellen, haben Sie eine Art »Stapelzeiger«. Sie können immer ablesen, wie hoch der Stapel gerade ist. So ähnlich funktioniert das auch beim Stapel im Computer. Natürlich werden dort keine Teller aufeinandergestapelt, sondern Zahlen. Und noch etwas anderes. Der Stapel wächst mit der Zahl der Einträge nicht nach oben, sondern nach unten. Dafür hat er aber mit dem Tellerstapel gemeinsam, daß es einiges an (Programmier-) Akrobatik erfordert, ein anderes als das oberste (auch wenn der Stack nach unten wächst, spricht man vom letzten als dem obersten Stapelelement) Stackelement zu entnehmen. Der Stapel ist 16-Bit-weise organisiert. Sie können also nur Doppelregister auf den Stack legen und von dort zurückholen. Dies hat den Vorteil, daß ein solcher Eintrag das gleiche Format hat wie eine Rücksprungadresse, was bei fortgeschrittenen Assembler-Kenntnissen eine Reihe interessanter Programmiermöglichkeiten auftut. Um ein Register auf den Stack zu schicken, gibt es den Befehl PUSH (auf Deutsch »Stoße« oder »Schiebe« auf den Stapel). Umgekehrt holt der Befehl POP den obersten Eintrag auf dem Stack in ein Doppelregister zurück. Der Begriff »Doppelregister« ist hier etwas anders als üblich definiert, denn er umfaßt nicht nur die Register BC, DE und HL, sondern auch den Akku gemeinsam mit dem Flag-Register. Dieses Register ist natürlich nicht im üblichen Sinn frei verwendbar. Es trägt den Kurznamen PSW, das heißt »Program Status Word«. Alle PUSH- und POP-Befehle finden Sie hier:

```
PUSH PSW      - POP PSW
PUSH B        - POP B
PUSH D        - POP D
PUSH H        - POP H
```

Eine wichtige Regel, deren Mißachtung meist zum »System-Crash« führt, sollten Sie als angehender Maschinencode-Programmierer nie vergessen. Da der Stack nicht nur Ihre Daten, sondern auch die Rücksprungadresse verwaltet, mit der der Computer wieder aus Ihrem Programm herausfindet, müssen Sie immer darauf achten, daß der Stack-Pointer vor einem Rücksprungbefehl genau denselben Wert hat wie beim Eintritt in das Programm oder Unterprogramm. Stackmanipulationen sollten Sie deshalb am Anfang noch den Profis überlassen.

Ebenso sollten Sie auch von Manipulationen des SP-Registers, die mit LXI SP, SPHL und XTHL möglich sind, anfangs die Finger lassen. Allgemein zur Verwendung empfohlen sind hingegen die folgenden Befehle, die den Datenaustausch des Prozessors mit dem Speicher ermöglichen:

LDA adresse lädt den Akkumulator mit dem Wert, der in der angegebenen Adresse gespeichert ist (Load Accumulator).

STA adresse macht genau das Umgekehrte wie LDA. Die Transportrichtung führt vom Akkumulator zur spezifizierten Speicherstelle (Store Accumulator).

LDAX B und LDAX D machen das gleiche wie MOV A, M, allerdings mit dem BC- und DE-Register. Sie laden den Inhalt der Adresse, auf die das Doppelregister zeigt, ins A-Register (Load Accumulator Using Extended Register).

STAX B und STAX D sind die Anpassung des MOV M,A-Befehls ans BC- und DE-Register und damit die Umkehrung des LDAX-Befehls (Store Accumulator Using Extended Register).

LHLD adresse lädt den Inhalt der Adresse ins HL-Register. Da das HL-Register 16 Bit breit ist, werden auch zwei Byte auf einmal übertragen (Load HL direct).

SHLD adresse ist die Umkehrung des LHLD-Befehls. Der Inhalt des 16-Bit-Registers HL wird an die angegebene Adresse und die direkt darauffolgende Speicherstelle geladen (Store HL direct).

Bisher noch nicht besprochen wurde die vorzeichenbehaftete Arithmetik: ADC (Add Register With Carry, »Addiere Registerinhalt zum Akku und berücksichtige das Carry-Flag«) und ACI (Add Immediate Value With Carry, »Addiere den unmittelbar im Programmtext folgenden Wert unter Berücksichtigung des Carry-Flags«). Die vorzeichenbehaftete Addition ist beim 8080-Prozessor nur mit 8-Bit-Länge möglich, wohingegen der Z80 auch 16-Bit-Addition ermöglicht. Vorzeichenlos ist aber die 16-Bit-Addition auch beim 8080-Chip möglich. DAD B, DAD D, DAD H und DAD SP addieren das jeweils angegebene Registerpaar zum »16-Bit-Akku« HL.

Die vorzeichenbehaftete Subtraktion ist mit SBB und SBI möglich. Diese beiden Befehle wirken – wie SUB und SUI – auf den 8-Bit-Akku. Die 16-Bit-Subtraktion wurde – wieder im Gegensatz zum Z80-Chip – gar nicht implementiert. Sie kann allerdings durch Negation des einen Werts und folgende Addition simuliert werden.

Ein sehr nützlicher Befehl heißt XCHG (Exchange, »Tausche«). Dieses Kommando tauscht die Inhalte der DE- und HL-Register gegeneinander aus.

Schiebung ist nicht strafbar

Die beiden letzten 16-Bit-Register-Befehle, die noch nicht behandelt wurden, heißen DCX und INX. INX B, INX D, INX H und INX SP erhöhen jeweils ein Doppelregister um Eins. DCX B, DCX D, DCX H und DCX SP vermindern das angesprochene Doppelregister um Eins.

Rotationsbefehle lassen die Bits im Akku »rotieren«. Zum Beispiel bedeutet RAL, daß jedes Bit im Akku um eine Position nach links geschoben wird:

```
Aus 0001 1010 wird
        0011 0100, da jedes Bit nach
                links bewegt wurde.
```

Das Bit, das aus dem Akku »herausgeschoben« wurde, geht nicht verloren. Es wird ins Carry-Flag gebracht und kann beispielsweise mit JC und JNC abgefragt werden. Das Carry-Bit, das vorher dort war, muß sich wieder hinten »anstellen« und wird zum letzten Bit im Akku (Bit 0). Nützlich ist der RAL-Befehl beispielsweise für eine schnelle Multiplikation mit 2, 4, 8, 16 und so weiter.

RAL heißt übrigens »Rotate Accumulator Left«, auf Deutsch »Rotiere den Akku nach links«. Folglich gibt es auch einen Befehl, der das Gegenteil macht. RAR rotiert den Akku nach rechts. Hier wandert das Bit rechts außen (Bit 0) ins Carry, und das Carry-Flag wird Bit 7 des Akkus. Sie sehen, daß man das Carry-Flag als neuntes Bit des Akkus betrachten kann.

Die beiden anderen Rotierbefehle, RLC und RRC, behandeln das Carry-Flag anders: Sie heißen »Rotate Left/Right Without Carry« oder »Rotiere

nach links oder rechts ohne das Carry-Flag«. RLC schiebt alle Bits um eine Stelle nach links, das Bit 7 wird Bit 0, wandert aber zusätzlich ins Carry-Flag. Der alte Inhalt des Carry geht verloren. RRC macht dasselbe, nur eben rechtsherum.

Natürlich kennen Sie jetzt noch nicht den gesamten 8080-Befehlssatz. Es gibt noch einige Kommandos, die nicht allzu häufig verwendet werden, aber doch recht nützliche Dinge tun:

STC (Set Carry Flag) setzt das Carry-Flag auf Eins. So kann zum Beispiel ein Unterprogramm dem Hauptprogramm Bericht erstatten, ob es seine Aufgabe ordnungsgemäß erfüllt hat.

CMC (Complement Carry Flag) komplementiert das Carry-Flag. Wenn es den Wert 0 hatte, wird es Eins; wenn er 1 war, ist er jetzt Null. Wollen Sie also das Carry-Flag löschen, müssen Sie die beiden Befehle kombinieren: STC und CMC.

PCHL (Load PC from HL) lädt den Programmzähler (das ist das Register, das dem Computer zeigt, an welcher Adresse er sich gerade befindet) mit dem Inhalt des HL-Registers. Auf deutsch: Der Computer ruft die Adresse auf, die im HL-Register angegeben ist.

DI (Disable Interrupts) und EI (Enable Interrupts) verbieten und erlauben die Abarbeitung von Interrupts. Befehle gleichen Namens gibt es auch in Basic.

IN port und OUT port werden Ihnen bei der Programmierung kaum begegnen, sofern Sie nicht vorhaben, direkt auf die Hardware zuzugreifen. Gerade das CPC-Betriebssystem stellt die ROM-Routinen für alle Eventualitäten zur Verfügung, so daß Sie auf IN und OUT sicher verzichten können.

Das war's dann

NOP (No operation): Dieser Befehl tut absolut nichts. Er bewirkt nur eine (sehr) kurze Wartezeit.

HLT (Halt) hält den Prozessor an, bis wieder ein Interrupt auftritt. Dieser Befehl wird aber sehr selten verwendet.

DAA (Decimal Adjust Accumulator) wird noch seltener gebraucht und dient dazu, BCD-Arithmetikroutinen anzusprechen. BCD (Binary Coded Decimal) ist eine besonders genaue Form der Zahlendarstellung und wird meist in kaufmännischen Programmen benötigt. Bei diesen darf es nicht vorkommen, daß der Computer aus 50.89 Mark etwa 50.8888888 Mark macht. Der Schneider besitzt keine BCD-Routine im ROM – leider!

Damit sind wir am Ende der Einführung in die 8080-Maschinensprache angelangt. Natürlich ist die Darstellung etwas gedrängt und knapp – andere Leute haben ganze Bücher über diesen Prozessor geschrieben. Aber Sie sollten zumindest die Grundlagen der Maschinencode-Programmierung mitbekommen haben. Sind Ihnen einige Sachen noch nicht ganz klar, empfehlen wir Ihnen, sich eines der zahlreichen 8080- und CP/M-Handbücher zu besorgen. Sie werden es sowieso zum Nachschlagen bei kniffligen Programmsituationen des öfteren benötigen! Was jetzt noch zu tun ist? Auch wenn es abgedroschen klingen mag: Übung macht den Meister. Um Maschinensprache zu verstehen, sollten Sie üben, üben und nochmals üben – möglichst an praktischen Beispielen. Und vergessen Sie nicht, je tiefer Sie in die Materie einsteigen, desto komplizierter wird es – aber auch umso interessanter. Basic ist recht leicht zu erlernen, stellt aber schon nach kurzer Zeit den Programmierer nicht mehr zufrieden. Maschinensprache ist dagegen ein unermeßliches Gebiet, dessen Grenzen Sie kaum erreichen dürften. (Martin Kotulla/hg)

Einstieg in CP/M – es lohnt sich



Zu jeder Diskettenstation liefert Schneider das Betriebssystem CP/M. Es eröffnet Ihrem Computer völlig neue Welten. Vorausgesetzt man weiß, wie es funktioniert.

Bekanntlich ist das Betriebssystem CP/M auf vielen Computern unterschiedlichster Hersteller lauffähig. Wie ist das möglich, wenn doch alle Geräte eine völlig voneinander abweichende Hardware aufweisen? Nun, ganz hardware-unabhängig ist CP/M natürlich auch nicht. Digital Research, die »Erfinderin« von CP/M, hatte aber die rettende Idee: Das Betriebssystem muß aus mehreren Teilen zusammengesetzt werden. Die direkt mit der Hardware zusammenhängenden Aufgaben erledigt das BIOS (Basic Input/Output System). Dieses BIOS führt die fundamentalen Aufgaben des Betriebssystems durch. Es gibt Zeichen auf dem Bildschirm aus, fragt die Tastatur ab, wählt das Diskettenlaufwerk aus und so weiter.

Das BIOS wird vom Hersteller des Computers für jedes Computermodell speziell entwickelt. Es verfügt über eine Tabelle, in die alle Einsprungadressen für die verschiedenen Routinen eingetragen sind. So kann das BDOS, das »Basic Disk Operating System« über diese genau definierten Adressen auf das BIOS zugreifen. Das BDOS wurde von Digital Research entwickelt und ist systemunabhängig. Auf allen CP/M 2.2-Computern (also auch auf den Schneider-Computern) ist es vom Programmcode her identisch. Schon auf dieser Stufe profitieren die Programmierer also von der Universität.

Die nächsthöhere Stufe sind schon die Anwenderprogramme, die der Benutzer in den Speicher lädt. Ein Beispiel verdeutlicht die Zusammenarbeit der Komponenten. Das Anwenderprogramm (zum Beispiel Wordstar) will ein Zeichen auf dem Bildschirm ausgeben. Dazu ruft es das BDOS auf, das den Auftrag dann an das BIOS weitervermittelt. Das BIOS gibt letztlich dann das Zeichen auf dem Monitor aus.

Bei den Schneider-Computern arbeitet das BIOS etwas anders. Es wäre völlig unsinnig, alle grundlegenden Funktionen neu zu programmieren, denn der Computer besitzt ja schon eine riesige Programmbibliothek – das normale Basic-Betriebssystem, mit dem Sie beim Einschalten des Computers in Berührung kommen. Also transformiert das BIOS beim Schneider wiederum den Befehl (in unserem Beispiel die Bildschirmausgabe) und gibt ihn an das normale Betriebssystem weiter. Und dieses führt nun endlich die Anweisung aus und bringt das Zeichen auf den Bildschirm.

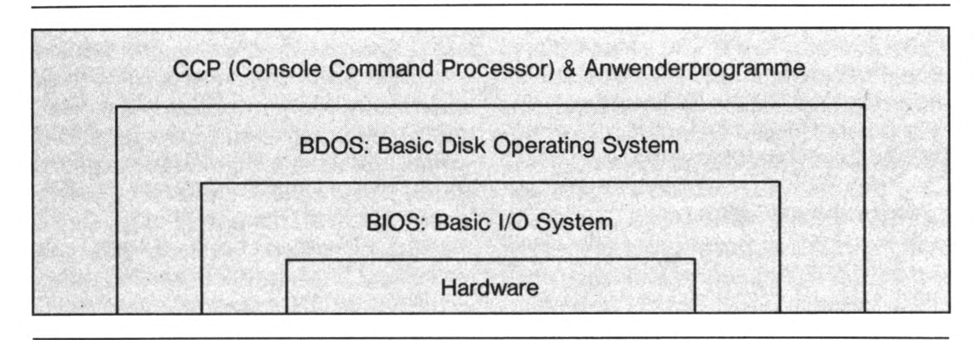


Bild 1. Die Struktur von CP/M im Überblick

Das BDOS führt aber nicht nur eine lineare 1:1-Umsetzung in BIOS-Aufrufe durch. Ein Großteil der BDOS-Befehle ist weit komplizierter aufgebaut. Man denke nur an die relative Dateiverwaltung (Datensätze im Direktzugriff), die unter CP/M ohne weiteres möglich ist. Bei diesen Kommandos muß das BDOS eine Menge Aufgaben selbst durchführen. Durch das BIOS kann es nämlich nur ganze Sektoren lesen oder beschreiben.

Jedes Betriebssystem benötigt letztlich eine Art »Benutzeroberfläche«. Der Bediener muß ja irgendwie von der Tastatur aus seine Befehle eingeben können. Dazu dient bei CP/M ein Programm mit dem Namen CCP (Console Command Processor). Dies ist der »Befehlsempfänger«, der von der Tastatur Kommandos entgegennimmt und in Betriebssystem-Aufrufe umsetzt. Solch ein Tastenbefehl ist zum Beispiel DIR. DIR beauftragt den CCP, das Inhaltsverzeichnis der Diskette auszugeben. Der Weg führt dabei wieder über das BDOS zum BIOS. So gesehen bestehen gewisse Ähnlichkeiten zwischen dem CCP und einem ganz normalen Anwenderprogramm.

Den schematischen Aufbau, die Struktur von CP/M 2.2, zeigt Bild 1. Man kann darin sehen, wie die einzelnen Programmteile immer komplexere Aufgaben erfüllen. Gleichzeitig muß permanent auf die darunterliegenden Programmteile zurückgegriffen werden.

Diese symbolische Darstellung hat allerdings nichts mit der tatsächlichen Speicherverteilung von CP/M zu tun. Bild 2 zeigt, wie die einzelnen Teile des Betriebssystems beim Schneider tatsächlich im Speicher angeordnet sind.

CCP, TPA – kein Geheimnis

In dieser Grafik tauchen einige Fachbegriffe auf, die wir bisher noch nicht besprochen haben. Zwischen 0000 und 00FF hex liegt die Zeropage des Betriebssystems. In diesem Speicherbereich (256 Bytes umfassend) »merkt« sich der Computer die wichtigsten Informationen und Einsprungstellen ins CP/M-System. Weiter unten werden wir die Zeropage wesentlich ausführlicher darstellen, da sie beim Programmieren unersetzlich ist.

Im Bereich 0100 bis 9700 hex befindet sich die TPA (Transient Programm Area). Dies ist der Speicherbereich, in den die eigentlichen Programme wie Textverarbeitungssysteme, Compiler oder Tabellenkalkulationen geladen werden. Von 9700 bis 9F06 hex liegt dann der bereits erwähnte CCP (Console Command Processor). Da er während der Bearbeitung eines Programms nicht benötigt wird, darf das Programm, das in der TPA liegt, ihn überschreiben. Nach dem Programmende wird er von der Diskette automatisch nachgeladen.

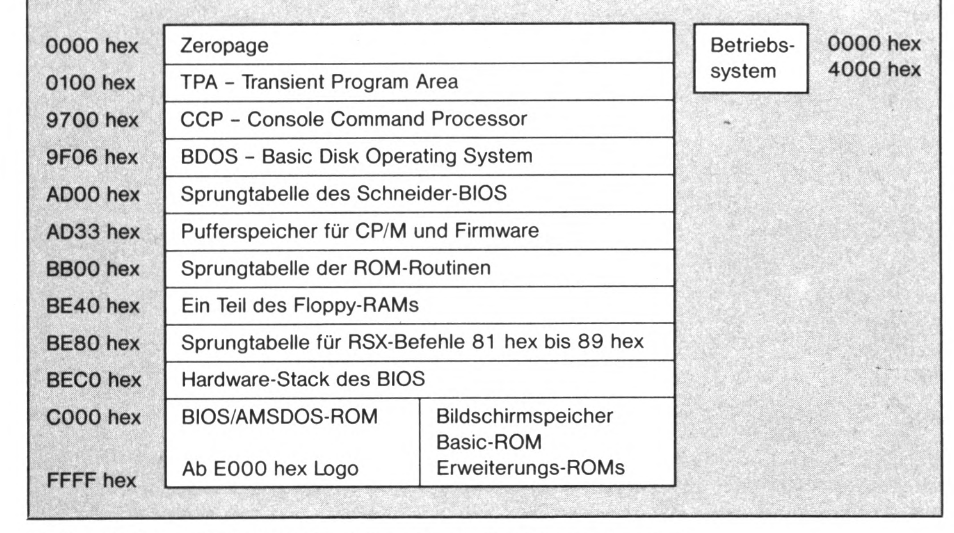
Im Adreßbereich von 9F06 bis AD00 hex steht das BDOS (Basic Disk Operating System). Zwischen AD00 und AD33 hex befindet sich die Sprungtabelle des Schneider-BIOS. Der Speicher von AD33 bis BB00 hex wird teilweise von CP/M und teilweise von der Schneider-Firmware belegt. Dort sind alle wichtigen Systeminformationen wie Bildschirmmodus und Farben, Tastaturtabellen und Belegung der Funktionstasten untergebracht, aber auch der Diskettenpuffer. Ab BB00 bis BE40 hex trägt der Computer nach dem Start automatisch die Sprungvektoren des Basic-Betriebssystems ein. Sie werden auch beim Laden von CP/M nicht überschrieben. So kann das BIOS über die standardisierten – und im Firmware-Handbuch dokumentierten – Schnittstellen ROM-Routinen aufrufen.

Zwischen BE40 bis BE80 hex findet man einen kleinen RAM-Speicher, der vom Floppy-ROM – genauso wie unter Basic – zur Speicherung wichtiger Daten benutzt wird. Hier finden der Disketten-Controller und das Betriebssystem Interrupt-Routinen zur Steuerung des Floppy-Laufwerks, Adreßzeiger auf weitere Speicherbereiche, die der Diskette zur Verfügung gestellt werden, Werte für Warteschleifen beim Laufwerk-Start und -Stop und andere für den Diskettenzugriff unerläßliche Informationen.

Direkter Diskettenzugriff

Der Speicher von BE80 bis BEC0 hex enthält für das BIOS eine Sprungtabelle. Diesmal sind es die Adressen der RSX-Befehle 81 bis 89 hex. Diese RSX-Kommandos sind von Basic aus nicht erreichbar, allenfalls durch Maschinencode-Programme. Sie stellen »niedere« Floppy-Routinen zur Verfügung, zum Beispiel zum Lesen, Schreiben und Formatieren von Sektoren. Von BEC0 bis C000 hex ist der Speicher für den Z80-Stack reserviert, den natürlich auch das Betriebssystem benötigt.

Etwas kompliziert wird es ab der Adresse C000 hex: Dort liegt das BIOS- und Amsdos-ROM, das sich beim CPC 464 in dem Kasten befindet, der auf den Computer hinten aufge-



Adresse	Inhalt	
0000 hex	Zeropage	
0100 hex	TPA – Transient Program Area	
9700 hex	CCP – Console Command Processor	
9F06 hex	BDOS – Basic Disk Operating System	
AD00 hex	Sprungtabelle des Schneider-BIOS	
AD33 hex	Pufferspeicher für CP/M und Firmware	
BB00 hex	Sprungtabelle der ROM-Routinen	
BE40 hex	Ein Teil des Floppy-RAMs	
BE80 hex	Sprungtabelle für RSX-Befehle 81 hex bis 89 hex	
BEC0 hex	Hardware-Stack des BIOS	
C000 hex	BIOS/AMSDOS-ROM Ab E000 hex Logo	Bildschirmspeicher Basic-ROM Erweiterungs-ROMs
FFFF hex		

Betriebssystem	
	0000 hex
	4000 hex

Bild 2. So ist CP/M 2.2 im Speicher des Schneiders abgelegt



steckt wird. Bei den anderen Schneider-Modellen ist dieses ROM hingegen auf der Hauptplatine angebracht. Parallel dazu adressiert der Prozessor aber auch den Bildschirmspeicher, das Basic-ROM und eventuelle Erweiterungs-ROMs. Für CP/M dauernd sichtbar ist allerdings nur das BIOS/Amsdos-ROM. Es hätte auch verheerende Folgen, wenn bei einem BIOS-Aufruf der Video-RAM-Bereich in den Adreßbereich eingeblendet wäre.

Die Betriebssystem-Routinen, die im ROM zwischen 0000 und 3FFF hex parallel zur TPA liegen, können aber das Floppy-ROM aus- und den Bildschirmspeicher einblenden, da sie vor der Rückkehr ins BIOS den alten ROM-Zustand wiederherstellen. Ab Adresse E000 hex befindet sich übrigens ein ROM-Programm, das mit der Diskettenstation gar nichts und mit dem CP/M-Betriebssystem nur entfernt zu tun hat: ein Teil des Logo-Interpreters.

Auf der Benutzerebene bemerkt man die »Innereien« nicht

Nach diesem Ausflug ins »Innenleben« des Schneiders kehren wir wieder auf die Benutzerebene, den CCP, zurück. Ins CP/M-Betriebssystem gelangen Sie, indem Sie eine Systemdiskette ins Laufwerk A einlegen und den RSX-Befehl **I**CPM benutzen. Sofort wird CP/M – mit dem CCP – geladen. Dieser versteht einige Kommandos, die zur Arbeit mit der Diskettenstation unerläßlich sind:

- **DIR** zeigt das Inhaltsverzeichnis der Diskette an.
- **ERA** löscht eine Datei oder Dateigruppe von der Diskette.
- **REN** benennt eine Datei um.
- **USER** wählt den Benutzerbereich aus.
- **TYPE** listet eine Diskettendatei auf dem Bildschirm.
- **SAVE** speichert die TPA oder Teile von ihr auf Diskette.
- **A:** wählt das Laufwerk A als Bezugslaufwerk, B die Floppy B.

Vor jede Eingabezeile schreibt der CCP den Buchstaben des verwendeten Diskettenlaufwerks und ein Größer-Zeichen – sozusagen als Aufforderung zur Eingabe von Befehlen. Diese beiden Zeichen werden oft auch als »Prompt« bezeichnet.

Die sieben beschriebenen Befehle heißen »resident«, weil sie dauernd im Computer »residieren« und damit ständig verfügbar sind. Vernünftig arbeiten läßt sich allein mit diesen Befehlen aber nicht. So gibt es zusätzlich sogenannte transiente Befehle. Das sind Programme, die bei jedem Aufruf erst von einer Diskette in die TPA geladen werden müssen. Sie besitzen als Dateierkennung den Zusatz ».COM«. Das heißt »Command File« (Kommandodatei). COM-Files sind alles, was unter CP/M als Programm fungiert: also nicht nur FORMAT.COM, PIP.COM, STAT.COM und die anderen Dienstprogramme, die Sie auf der Systemdiskette finden, sondern auch WS.COM (Wordstar), TURBO.COM (Turbo-Pascal) und andere Software. Der CCP macht keinen Unterschied zwischen dem Aufruf residenter und transienter Befehle. Geben Sie DIR ein, erkennt der CCP diesen Befehl als resident und führt ihn unmittelbar aus. Tippen Sie aber »WS« für Wordstar, sucht der CCP ihn erst einmal bei den residenten Befehlen. Findet er ihn dort nicht, sucht er deshalb auf der Diskette weiter. Ist er endlich fündig geworden, lädt er das Programm und startet es. Ist kein Programm dieses Namens auf der Diskette, meldet sich der CCP mit »WS« zurück.

Allzu informativ kann man die Fehlermeldungen von CP/M zwar nicht nennen, man muß aber bedenken, daß CP/M zu einer Zeit entwickelt wurde, als Speicherplatz noch so kostbar war, daß man um jedes Byte kämpfte.

Einige Funktionen des CCP sind so wichtig, daß sie auf einen einzigen Tastendruck hin ausgelöst werden. Dies geschieht durch Drücken der Control-Tasten zusammen mit einem Kennbuchstaben. Heißt es also »Control-A«, »CTRL-A« oder ganz kurz »↑A«, bedeutet dies, daß Sie die CTRL-Taste gleichzeitig mit dem entsprechenden Buchstaben drücken sollen. Es gibt in CP/M folgende Controlcodes:

CTRL-C löst einen Warmstart des Computers aus – das Betriebssystem lädt den CCP neu von der Diskette. Wichtig ist diese Tastenkombination, wenn Sie die Diskette im Laufwerk tauschen wollen. Bei jedem Diskettenwechsel müssen Sie CTRL-C drücken. Sonst könnte folgendes passieren: Der Computer arbeitet noch mit einer offenen Datei. Sie wechseln aber die Diskette mit dieser Datei gegen eine neue aus, und der Computer schreibt die weiteren Datensätze und den Directory-Eintrag auf die neue Diskette. Sie hätten dann im schlimmsten Fall zwei unbrauchbare Disketten. Da Digital Research dieses Problem erkannte, verlangt CP/M bei jedem Diskettenwechsel ein CTRL-C. Andernfalls erhalten Sie beim nächsten Schreibzugriff,

CP/M reagiert auf Bedienerfehler manchmal empfindlich

zum Beispiel mit ERA, die Fehlermeldung »Bdos Err On A: R/O«. Dies bedeutet »BDOS-Fehler auf Laufwerk A, nur Lesen erlaubt«. Nach einem Diskettenwechsel erklärt CP/M die neue Diskette kurzerhand bis zum nächsten Warmstart als schreibgeschützt – um Fehler zu vermeiden. Ärgerlich ist es aber, wenn man mitten aus einem Programm wegen solch einer BDOS-Fehlermeldung herausgeworfen wird. Um Fehlbedienungen zu vermeiden, ist CTRL-C übrigens nur wirksam, wenn Sie es an der ersten Stelle einer Zeile eingeben.

CTRL-E erlaubt Ihnen, Ihre Eingabe in der folgenden Zeile fortzusetzen, beispielsweise

A>DIR ↑ E

B: (Enter)

CTRL-H löscht das letzte eingegebene Zeichen und ist damit gleichwertig mit der DEL-Taste.

CTRL-I bewirkt einen Sprung zur nächsten Tabulator-Position. Stattdessen kann man auch die TAB-Taste drücken.

CTRL-J ist gleichwertig mit ENTER. Die Taste schließt die Eingabezeile ab.

CTRL-M besitzt ebenfalls die gleiche Funktion wie Enter.

```
A>DIR
A: MOVCPM   COM : PIP      COM : SUBMIT   COM : XSUB     COM
A: ED       COM : ASM      COM : DDT      COM : LOAD     COM
A: STAT     COM : DUMP     COM : DUMP     ASM : AMSDOS   COM
A: FILECOPY COM : SYSGEN   COM : BOOTGEN  COM : COPYDISC COM
A: CHKDISC  COM : DISCCOPY COM : DISCCHK  COM : SETUP    COM
A: FORMAT   COM : CSAVE    COM : CLOAD    COM : EX1      BAS
A: EX2      BAS : ROINTIME DEM


A>DIR
A: LOGO     COM : SETUP    COM : AMSDOS   COM
A>
```

Bild 3. Das Directory der beiden Seiten der Systemdiskette des Schneider CPC 464 und 664



CTRL-P: Alle Bildschirmausgaben werden auf dem Drucker mitprotokolliert, bis Sie wieder CTRL-P drücken. Achten Sie aber darauf, daß der Drucker empfangsbereit ist, da der Schneider sich sonst in einer Endlosschleife festfährt. Andere Eingaben als CTRL-SHIFT-ESC sind dann nicht mehr möglich. Diese drei Tasten lösen aber einen Neustart aus, der alle Daten und Programme im Speicher vernichtet.

CTRL-R wiederholt die Eingabezeile auf dem Bildschirm. Diese Funktion ist nur für ältere Terminals interessant, die auf dem Bildschirm keine Zeichen löschen können.

CTRL-S unterbricht die Textausgabe auf dem Bildschirm, bis eine andere Taste (bei CP/M 3.0 muß es wieder CTRL-S sein) gedrückt wird. Anwendung findet CTRL-S häufig beim Befehl TYPE.

CTRL-X löscht die eingetippte Befehlszeile vom Bildschirm.

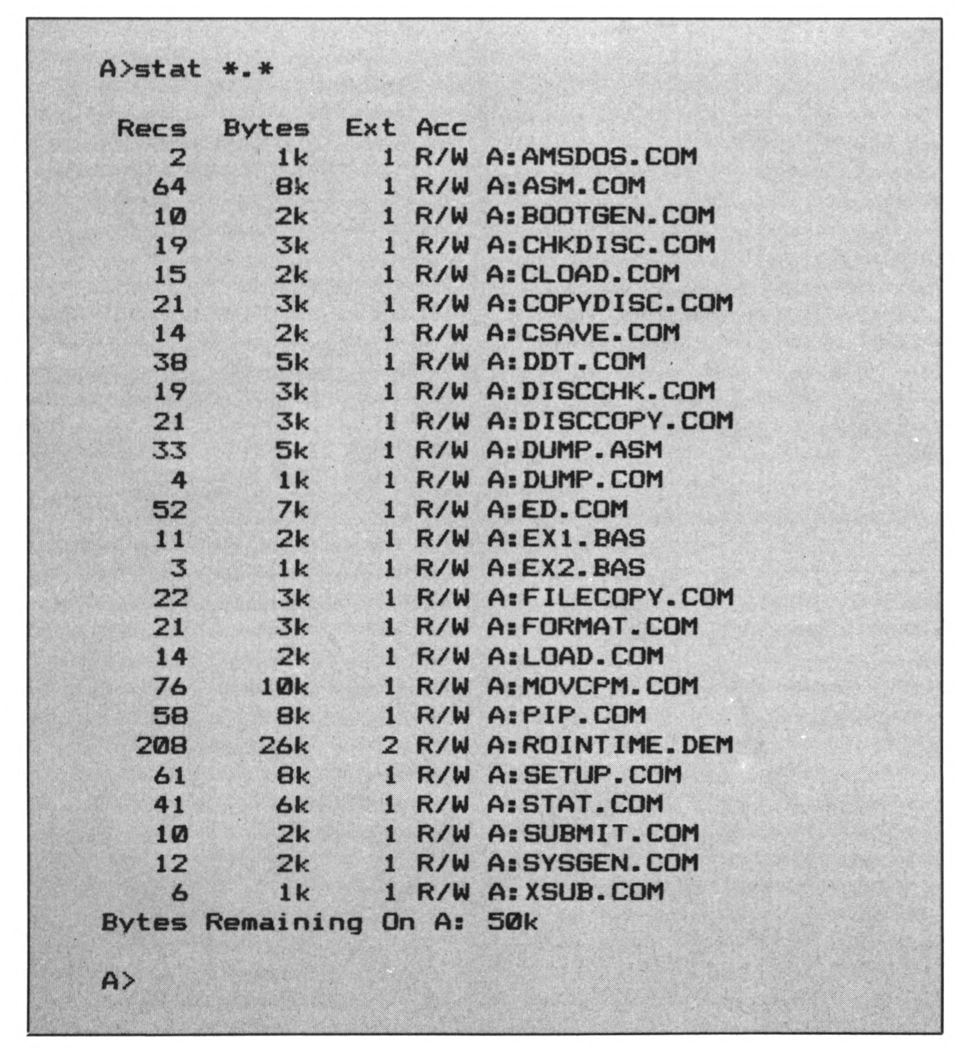
CTRL-U bewirkt, daß die eingetippte Zeile vom CCP ignoriert wird. CTRL-U ist zum Beispiel bei Fehleingaben zu empfehlen und in etwa mit der ESC-Taste unter Basic vergleichbar.

CTRL-V gemeinsam mit der ENTER-Taste läßt der CCP die eingetippte Zeile wie bei CTRL-U ignorieren. Sie wird aber trotzdem nochmals auf dem Bildschirm angezeigt. CTRL-U ist jedoch vorzuziehen, da CTRL-V gleichzeitig ein Bildschirmsteuerzeichen ist, das den Transparentmodus einschaltet. In diesem werden Zeichen nicht mehr gelöscht, sondern übereinander gedruckt.

Es ist übrigens ohne Bedeutung, ob Sie Ihre Eingaben in Klein- oder Großbuchstaben machen, denn der CCP wandelt alles in Großbuchstaben um.

Kostenlose Helfer

Sie haben sich sicher schon einmal das Inhaltsverzeichnis der mitgelieferten CP/M-Systemdiskette ausgeben lassen. Wenn nicht, zeigt Bild 3 das Directory beider Diskettenseiten. Für die Arbeit mit CP/M ist nur die Seite A interessant, denn dort befinden sich alle unentbehrlichen Hilfsprogramme. Man kann zwei Gruppen von Programmen unterscheiden: die CP/M-Standardprogramme und die speziell auf den Schneider zugeschnittene Software. Letztere umfaßt die Programme AMSDOS (Rückkehr aus CP/M nach Basic), FILECOPY (Kopieren von Dateien mit einem Laufwerk), COPYDISC (Kopierroutine (sektorweise) für zwei Laufwerke). DISCOPY (Kopierroutine (sektorweise) für eine Floppy) sowie DISCCHK und CHKDISC zum Vergleich zweier Disketten nach dem Kopieren (Verify). Dazu kommen noch SETUP (Einstellen der CP/M-Systemparameter wie Tastaturbelegungen und Initalisierungsdaten der Schnittstellen), FORMAT (Formatieren von Disketten), CSAVE und CLOAD (Speichern von CP/M-Dateien auf Kassette und umgekehrt). Außerdem meldet das Inhaltsverzeichnis noch drei Dateien, die nichts mit CP/M, sondern mit Basic zu tun haben: ROINTIME (Demonstration eines Spielprogramms), EX1 und EX2 (Speichern von Grafiken auf Diskette).



```
A>stat *.*

 Recs  Bytes  Ext Acc
    2     1k    1 R/W A:AMSDOS.COM
   64     8k    1 R/W A:ASM.COM
   10     2k    1 R/W A:BOOTGEN.COM
   19     3k    1 R/W A:CHKDISC.COM
   15     2k    1 R/W A:CLOAD.COM
   21     3k    1 R/W A:COPYDISC.COM
   14     2k    1 R/W A:CSAVE.COM
   38     5k    1 R/W A:DDT.COM
   19     3k    1 R/W A:DISCCHK.COM
   21     3k    1 R/W A:DISCCOPY.COM
   33     5k    1 R/W A:DUMP.ASM
    4     1k    1 R/W A:DUMP.COM
   52     7k    1 R/W A:ED.COM
   11     2k    1 R/W A:EX1.BAS
    3     1k    1 R/W A:EX2.BAS
   22     3k    1 R/W A:FILECOPY.COM
   21     3k    1 R/W A:FORMAT.COM
   14     2k    1 R/W A:LOAD.COM
   76    10k    1 R/W A:MOVCPM.COM
   58     8k    1 R/W A:PIP.COM
  208    26k    2 R/W A:ROINTIME.DEM
   61     8k    1 R/W A:SETUP.COM
   41     6k    1 R/W A:STAT.COM
   10     2k    1 R/W A:SUBMIT.COM
   12     2k    1 R/W A:SYSGEN.COM
    6     1k    1 R/W A:XSUB.COM
Bytes Remaining On A: 50k

A>
```

Bild 4. Informationen über die Programme der Systemdiskette bekommen Sie mit dem Befehl »STAT*.*«

Wichtige Utilities

Die anderen Programme wurden von Digital Research entwickelt und gehören zum Lieferumfang von CP/M 2.2. Sie lassen sich in zwei Kategorien unterteilen: in Routinen, mit denen sich jeder Benutzer auseinandersetzen muß, und in andere, die speziell für Programmierer gedacht sind. Leider hat Schneider im Handbuch der Diskettenstation nicht allzu viel Platz für die Beschreibung der Dienstprogramme aufgewendet; holen wir es also nach:

SYSGEN (Generate System Sector) und **BOOTGEN** (Generate Boot Sector) benötigen Sie, wenn Sie eine im Vendor-Format beschriebene Diskette zu einer CP/M-Diskette machen wollen. Die beiden Programme schreiben die Systemspuren auf die neue Diskette.

MOVCPM (Move CP/M System) erlaubt Ihnen, eine veränderte CP/M-Version auf die Diskette zu schreiben. Dies kann notwendig werden, wenn Sie beispielsweise Speicherplatz fest reservieren und damit das Betriebssystem im Speicher nach unten verschieben müssen. Allerdings wird dadurch der sowieso schon recht magere Umfang der TPA von rund 39 KByte noch knapper. Will man eine Speichererweiterung einbauen, so läßt sich mit MOVCPM die TPA anpassen.

STAT (Status) ist ein Allzweck-Programm, das den freien Speicherplatz auf der Diskette angibt und meldet, ob eine Diskette schreibgeschützt ist. Eine typische Ausgabe lautet:

»A: R/W, Space: 50 k«

»STAT A:« oder »STAT B:« meldet, wieviel Speicherplatz auf dem angegebe-

nen Laufwerk noch frei ist. »STAT filename.ext« liefert Informationen über die angegebene Datei. Bei »STAT *.*« meldet das Programm diese Informationen über alle Dateien auf der Diskette. Es werden die Programmlänge in Records und Kilobytes, die Zahl der Extents (Directory-Einträge), der Dateistatus (R/W = Read/Write, Lesen und Schreiben erlaubt; R/O = Read-Only, nur Lesen erlaubt) und der Dateiname ausgegeben. Einen Ausdruck der Systemdiskette zeigt Bild 4.

»STAT filename.ext $SYS« erklärt die angegebene Datei zur »Systemdatei«. Diese wird im Directory nicht mehr gelistet.

»STAT filename.ext $DIR« verwandelt eine Systemdatei wieder in ein normales File, das auch im Inhaltsverzeichnis auftaucht.

»STAT filename.ext $R/O« schützt eine Datei dauerhaft gegen Schreibversuche. Dies kann mit »STAT filename.ext $R/W« wieder rückgängig gemacht werden.

»STAT A:=R/O« und »STAT B:=R/O« erklären das Laufwerk bis zum nächsten Warmstart (CTRL-C) als schreibgeschützt.

»STAT USR:« liefert die aktuelle User-Nummer und teilt dem Benutzer außerdem mit, welche User-Bereiche auf der Diskette benutzt werden.

»STAT DEV:« meldet die Zuordnung der logischen Peripheriegeräte (Devices) zu den physikalischen Geräten. Es gibt vier logische Geräte:

»CON« die Konsole,

»RDR« der Lochstreifenleser (Reader),

»PUN« der Lochstreifenstanzer (Puncher)

»LST« der Druckerausgang (Lister).

Rein und raus

Beim Schneider finden normalerweise nur die Bezeichnungen »LST:« und »CON:« Verwendung. Dabei ist die Bezeichnung »CON:« eigentlich doppeldeutig. Als Quelle ist damit die Tastatur gemeint, als Ziel der Bildschirm. Schließlich können Sie – zumindest nach CP/M-Verständnis – keine Zeichen vom Bildschirm lesen oder an die Tastatur schreiben.

»STAT log:=phy:« ordnet einem logischen Ausgabegerät ein physikalisches zu, so wie es dann von »STAT DEV:« gemeldet wird. Damit ließen sich zum Beispiel serielle Schnittstellen beim Schneider ins System integrieren. Welche physikalischen Geräte sich den logischen zuordnen lassen, zeigt Bild 5.

»STAT DSK:« meldet genaue Informationen über die Formatierungseigenschaften der Disketten. Statt »STAT DSK:« können Sie auch »STAT A:DSK:« oder »STAT B:DSK:« eingeben, um ein spezielles Laufwerk anzusprechen.

»STAT VAL:« zeigt eine Auflistung aller gültigen Befehlsformate des STAT-Programms. In Bild 6 sehen Sie einen Ausdruck davon.

PIP (Peripheral Interchange Program) ist ein sehr leistungsfähiges Programm zum Austausch von Daten zwischen den verschiedenen Peripheriegeräten. Es gibt zwei »Betriebsarten« für das PIP-Programm. Entweder geben Sie direkt nach »PIP« die gewünschte Operation an oder Sie gelangen in einen Eingabemodus, bei dem Sie wiederholt Befehle geben können, ohne die Routine dauernd neu laden zu müssen. Wollen Sie beispielsweise die beiden Dateien PROG.A und DATEI.B von Laufwerk A auf Laufwerk B kopieren, können Sie eingeben:

```
A)PIP B:=A:PROG.A
A)PIP B:=A:DATEI.B
```

Im Eingabemodus geht das etwas einfacher und vor allen Dingen schneller:

```
A)PIP
*B:=A:PROG.A
*B:=A:DATEI.B
*↑C
```

Logisches Gerät CON:	
CRT:	Cathode Ray Tube (Bildschirm)
TTY:	Teletype (Fernschreiber)
BAT:	Batch (Stapelverarbeitung für Lochkarten)
UC1:	User Console 1 (vom Benutzer festlegbar)
Logisches Gerät RDR:	
TTY:	Teletype (Fernschreiber)
PTR:	Paper Tape Reader (Lochstreifenleser)
UR1:	User Reader 1 (vom Benutzer festlegbar)
UR2:	User Reader 2 (vom Benutzer festlegbar)
Logisches Gerät PUN:	
TTY:	Teletype (Fernschreiber)
PTP:	Paper Tape Puncher (Lochstreifenstanzer)
UP1:	User Puncher 1 (vom Benutzer festlegbar)
UP2:	User Puncher 2 (vom Benutzer festlegbar)
Logisches Gerät LST:	
TTY:	Teletype (Fernschreiber)
CRT:	Cathode Ray Tube (Bildschirm)
LPT:	Line Printer (Zeilendrucker)
UL1:	User Lister 1 (vom Benutzer festlegbar)
Zusätzliche PIP-Gerätenamen:	
PRN:	Printer Entspricht LST: mit der Option [PNT8]
NUL:	Nulls Entspricht PUN:, sendet aber 40 Nullbyte als Vor- und Nachlauf
EOF:	End-Of-File Enspricht PUN:, erzeugt aber am Dateiende die Kennung Control-Z
INP:	User Input Vom Benutzer derfinierbares Gerät
OUT:	User Output Vom Benutzer definierbares Gerät

Bild 5. So ordnet CP/M die logischen Peripheriegeräte den tatsächlichen Einheiten zu

```
A>stat val:

Temp R/O Disk: d:=R/O
Set Indicator: d:filename.typ $R/O $R/W $SYS $DIR
Disk Status  : DSK: d:DSK:
User Status  : USR:
Iobyte Assign:
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:
A>
```

Bild 6. STAT über sich selbst. Der Befehl »STAT VAL:«

Sie können mit PIP-Dateien von allen möglichen Peripheriegeräten zu anderen Geräten übertragen:

»PIP B:=A:*.*« kopiert die ganze Diskette von Laufwerk A nach B.

»PIP A:P.COM=B:Q.CON« kopiert Q.COM von B nach A und nennt die neue Datei P.COM.

»PIP TEXT.TXT=COM:« liest Eingaben von der Tastatur in die Diskettendatei TEXT.TXT, bis Sie CTRL-Z drücken.

»PIP LST:=CON:« nimmt Tastatureingaben an und überträgt sie an den Drucker, bis Sie CTRL-Z drücken. Um hier in eine neue Zeile zu gelangen, müssen Sie neben ENTER auch noch CTRL-J (Line Feed) eingeben.

Text mit Zeilennummern

Es gibt spezielle Befehle, die Sie PIP beim Übertragen der Dateien mitgeben können, beispielsweise zum automatischen Einfügen von Zeilennummern oder Löschen des 7. Bits aller Zeichen. Diese Anweisungen werden nach den Dateinamen in eckigen Klammern angegeben. Ein Befehl heißt zum Beispiel »[N]«. Er fügt bei der Dateiausgabe Zeilennummern ein. Wollen Sie also eine Textdatei »B:FILE.TXT« auf dem Drucker mit Zeilennummern ausgeben, erreichen Sie das durch »PIP LST:=B:FILE.TXT[N]«. Auch die Kombination mehrerer solcher Befehls-Buchstaben enthält die Tabelle in Bild 7.

SUBMIT und **XSUB** sind Programme, die Ihnen eine Menge unnötiger Tipparbeit abnehmen. Normalerweise geben Sie die Befehle des CCP über die Tastatur ein. Sie können aber auch alle Befehle in eine Datei mit der Endung .SUB, zum Beispiel »COPY.SUB«, schreiben und dann durch »SUBMIT« ausführen lassen. Der Computer verhält sich dann so, als kämen die Eingaben von der Tastatur, holt sie aber in Wirklichkeit von der Diskettendatei. Bei regelmäßig wiederkehrenden Routineaufgaben müssen Sie dann nicht vor dem Computer sitzen, um immer wieder neue Befehle einzugeben – SUBMIT macht das für Sie automatisch.

Programmieren Sie zur Übung doch einmal eine kleine Datei. Wenn Sie keinen Editor zur Verfügung haben, können Sie PIP verwenden: »PIP A:INFO.SUB=CON:«. Alle Tastatureingaben werden jetzt in die Datei »INFO.SUB« übertragen. Beenden können Sie Ihre Eingabe mit CTRL-Z. Hier der Dateiinhalt für unser Beispiel:

```
A:
DIR
DIR B:
STAT *.*
STAT DSK:
STAT USR:
```

Wenn Sie nur ein Diskettenlaufwerk besitzen, lassen Sie die Zeile »DIR B:« weg. Sobald Sie die Datei angelegt haben, müssen Sie sicherstellen, daß sich SUBMIT auf der eingelegten Diskette befindet. Tippen Sie dann »SUBMIT INFO« ein und Sie erhalten eine Menge Informationen über Disketten und Laufwerk.

Fast ein Programm

Auch Stringvariablen werden von SUBMIT verarbeitet. Diese heißen $1 bis $9. Da hinter »SUBMIT INFO« auch Texte eingegeben werden können, werden diese automatisch den Strings zugeordnet. Lautet Ihre Eingabe etwa:

»SUBMIT PROG DATEI1.ASM FILE2.COM PROGRAMM3.BAS«

Dann erkennt der Computer SUBMIT als den Programmnamen und PROG als »PROG.SUB«, die Submit-Datei. Der Stringvariablen $1 wird »DATEI1.ASM« zugeordnet, $2 wird mit »FILE2.COM« und $3 mit »PROGRAMM3.BAS« belegt.

Legen Sie nun eine neue Datei unter dem Namen »FILE.SUB« an:

```
DIR  $1
STAT $1
TYPE $1
PIP  $2=$1
```

Um das Programm zu starten, ist diese Befehlsfolge notwendig: »SUBMIT FILE datei1.typ datei2.typ« (beispielsweise »SUBMIT FILE TEXT.TXT BRIEF.WS«).

SUBMIT »übersetzt« dann diesen Aufruf so:

```
DIR TEXT.TXT
STAT TEXT.TXT
TYPE TEXT.TXT
PIP BRIEF.WS=TEXT.TXT
```

Die Datei TEXT.TXT haben Sie noch von der Erklärung des PIP-Kommandos auf Ihrer Diskette. Wenn Sie versuchen, auch in einem aufgerufenen Programm die Tastatureingabe durch eine Submit-Datei zu simulieren, werden Sie bemerken, daß das so nicht geht. Denn SUBMIT arbeitet nur auf der CCP-Kommandoebene. Für solch einen Zweck benötigen Sie XSUB. XSUB muß dazu ebenfalls in die SUB-Datei geschrieben werden. So können Sie zum Beispiel

Option		Bedeutung
B	=	Block Mode Transfer Kopieren, bis ein Control-S auftritt
Dx	=	Delete From Column x Löscht alle Zeichen ab der Spalte x
E	=	Echo Alle Zeichen werden auf dem Bildschirm ausgegeben
F	=	Delete Form Feeds Löscht alle ASCII-12-Zeichen (Form Feed) aus der Datei
Gx	=	Get From User Area x Holt die Datei aus dem Benutzerbereich x
H	=	Copy Intel-Hex File Stellt fest, ob die Datei eine Intel-Hex-File ist.
I	=	Ignore :00 Records Ignoriert alle Aufzeichnungen einer Intel-Hex-Datei, die mit :00 beginnen.
L	=	Lower Case Gibt die Datei in Kleinschrift aus.
N	=	Line Numbers Fügt vor jeder Zeile durchgehende Zeilennummern ein.
N2	=	Line Numbers (2) Wie [N], jedoch Zeilennummern mit führenden Nullen.
O	=	Copy Object File Kopiert Dateien, ohne das EOF-Zeichen Control-Z zu beachten.
Px	=	Pages Alle x Zeilen wird ein Form-Feed-Zeichen eingefügt.
Qtext↑S	=	Quit Copy Kopiert die Datei, bis der String »text« entdeckt wird.
R	=	Read System File Kopiert Systemfiles ($SYS)
Stext↑S	=	Start Copy Kopiert die Datei ab der Stelle, an der »text« gefunden wird.
Tx	=	Set Tabulators Setzt Tabulatoren alle x Stellen.
U	=	Upper Case Alle Buchstaben werden in Großbuchstaben umgewandelt.
V	=	Verify Data Überprüft bei Diskdateien die geschriebenen Zeichen nochmals.
W	=	Write Over R/O Files Überschreibt schreibgeschützte Dateien ohne Warnung.
Z	=	Set Bit 7 To Zero Löscht in allen Zeichen das siebte Bit.

Bild 7. Die Optionen von PIP

die PIP-Kommandoebene aufrufen, um ein Programm über die Tastatur einzugeben:

```
XSUB
PIP
XTEST.TXT=CON:
```

Die Routine geben Sie mit »PIP XSUBTEST.SUB=CON:« ein. Nach »SUBMIT XSUB TEST« startet das Programm PIP und geht zur Eingabe über. Ohne XSUB würde der Computer nach dem Start von PIP stehen bleiben, da keine Parameter (für PIP) mit übergeben werden.

Für Maschinensprache-Fans: ASM, LOAD und DDT

Das Dreigespann ASM, LOAD und DDT ist bereits alles, was ein Maschinensprache-Programmierer unter CP/M an Software braucht. Bei ASM handelt es sich um einen einfachen 8080-Assembler, LOAD übersetzt die vom Assembler gelieferten Dateien in ausführbare Programme und DDT ist ein recht mächtiger Maschinensprache-Monitor und Debugger.

Der DDT (Dynamic Debugging Tool) wurde bereits im zweiten Schneider-Sonderheft ausführlich erklärt. Falls Sie diese Ausgabe nicht zur Hand haben, finden Sie in Bild 8 eine Auflistung der DDT-Befehle. Sie sollten aber noch wissen, daß der DDT-Debugger prinzipiell mit Hexadezimal-Zahlen arbeitet. 3000 ist für ihn also nicht 3000 dez, sondern 3000 hex oder 12288 dez.

Der Assembler: nicht super, aber ausreichend

Auch wenn ASM (aufgerufen mit »ASM Quelldatei ASM«) nicht alle Spezialitäten hochentwickelter Assembler wie Makros oder Erzeugung relokatibler (verschiebbaren) Codes bietet, reicht er für unsere Zwecke vollauf aus. Er übersetzt alle 8080-Befehle, und das ist ja das Wichtigste. Zudem besitzt ASM neben dem Prozessorbefehlssatz noch folgende Pseudokommandos:

EQU ordnet einem Label einen Wert zu, zum Beispiel RESET EQU 0. Nach dieser Zuweisung kann RESET gleichwertig neben 0 verwendet werden:

```
RESET EQU 0
      CALL RESET
      LHLD RESET
```

Solche Marken dürfen auch vor Maschinencode-Befehlen stehen und besitzen dann als Wert die gerade aktuelle Adresse:

```
START LXI SP,0100H
```

(das H steht für hex).

Ein Beispiel:

Stellen Sie sich vor, daß der LXI-Befehl an der Adresse 0200 hex steht. START entspricht dann 0200 hex.

SET funktioniert ähnlich, ermöglicht aber wiederholte Definitionen des gleichen Labels:

```
X SET 2000H
LDA X
X SET 3000H
STA X
```

Diese Marken dürfen nicht den gleichen Namen haben wie ein Pseudokommando oder ein Maschinenbefehl (beispielsweise ORG oder RET). Die Labels können bis zu 16 Buchstaben lang sein. Dabei sind alle Stellen gültig. Außer Buchstaben und Zahlen ist als Teil eines Labels nur das Dollar-Symbol ($) erlaubt – zum Beispiel: RESET $CPM, WARM$BOOT. Es zählt aber nicht zu den 16 erlaubten Stellen.

SET und EQU dürfen auf bereits definierte Labels zurückgreifen, beispielsweise »RESET EQU 0« und »BDOS EQU RESET+9700H«. Hier sehen Sie auch, daß Berechnungen in den vier Grundrechenarten sowie die Logikfunktionen **AND, OR, XOR** und **NOT** sowie **SHL** (Shift Left, wie »2000H SHL 2«) und **SHR** (Shift Right, zum Beispiel »3000H SHR 4«) erlaubt sind. Aber Achtung: Diese Berechnungen werden nur während des Assemblierungsvorgangs durchgeführt, nicht während des Ablaufs des Maschinenprogramms. Sie sind damit kein Ersatz für die 8080-Arithmetikbefehle wie DCR, ADI, ORA oder SBB und eignen sich nicht für Rechnungen, bei denen Werte erst während der Programmbearbeitung zur Verfügung stehen.

Zwei weitere Pseudobefehle heißen **ORG (Origin)** und **END**. ORG setzt den Programmzähler auf den angegebenen Wert. An diese Adresse wird das zukünftige Programm geladen. Da die TPA bei 0100 hex beginnt, sollte die erste Zeile des Programms ORG 0100H lauten. END zeigt dem Assembler das Ende der Quelldatei an.

Auch eine bedingte Assemblierung ist gestattet. **IF** und **ENDIF** sind die Befehle dazu. So kann man Programmteile übersetzen lassen oder von der Assemblierung ausschließen, wenn eine angegebene Bedingung zutrifft.

Bedingte Assemblierung

Datenbereiche werden mit **DS** (Define Storage), **DB** (Define Byte) und **DW** (Define Word) in den Programmcode eingefügt:

DS 23 reserviert 23 Byte Speicherplatz, die vom Maschinencode-Programm zum Beispiel für Daten oder als Zwischenspeicher benutzt werden können. Der Inhalt dieses Bereichs ist undefiniert, also nicht zwangsläufig mit Nullen gefüllt. »DB 24.20H,010B,'A'« legt im Speicher die Bytes 24, hex 20, bis 010 und 65 (ASCII-Wert von A) ab. DW macht dasselbe für 16-Bit-Werte: »DW 16384,9000H«.

Der aktuelle Programmzähler (die Adresse, an der das Programm sich gerade befindet) wird durch das Dollar-

Befehl	Funktion / Syntax
A = Assemble	Mnemonics eingeben und assemblieren A [Adresse], Beispiel: A0100
D = Dump	Speicherbereich hexadezimal und ASCII ausgeben D [Von], [Bis], Beispiel: D0100,0200
F = Fill	Speicherbereich füllen F [Von], [Bis], [Womit], Beispiel: F3000,4000,E5
G = Go	Maschinenprogramm aufrufen, auch mit Breakpoints G [Wohin] oder G [Wohin], [Stop1], [Stop2], Beispiel: G0000
H = Hexcalc	Hexadezimale Addition und Subtraktion H [Zahl1], [Zahl2], Beispiel: H0003,1F3A
I = Input	Dateinamen zum Datei-Einlesen vorgeben I [Dateiname.Ext], Beispiel: IPROG.COM
L = List	Disassemblier-Funktion L [Von], [Bis], Beispiel: L9700,9800
M = Move	Speicherbereiche verschieben M [Von], [Bis], [Wohin], Beispiel: M9800,9F00,4000
R = Read	Datei in den Speicher lesen R [Offset], Beispiel: R0000
S = Store	Speicherbereiche hexadezimal abändern S [Adresse], Beispiel: S4000
T = Trace	Einzelschritt-Abarbeitung mit Ausgabe der CPU-Register T oder T [Schritte], Beispiel: T20
U = Untrace	Wie Trace, jedoch ohne Ausgabe der CPU-Register U oder U [Schritte], Beispiel: U20
X = Examine	CPU-Register anzeigen und abändern X oder XA, XB, XD, XH, XS, XP oder XC/XZ/XM/XE/XI

Bild 8. Keine Unglückszahl: 13 Befehle besitzt der Debugger DDT



zeichen repräsentiert. Um zum Beispiel zehn Byte vorwärts zu springen, braucht man nur JMP $+10 eingeben.

ASM erlaubt es, mehrere Befehle in einer Zeile unterzubringen. Ähnlich dem Doppelpunkt, der in Basic Befehle trennt (ein Beispiel: CLS:LIST), wird hier das Ausrufezeichen verwendet (CMP H ! RZ). Kommentare leitet – wie bei Z80/8080-Assemblern üblich – ein Strichpunkt ein. Aber auch den Stern (*), den manche Assembler vorziehen, akzeptiert ASM als Kommentarkennung:

```
; So wird kommentiert
* So auch
```

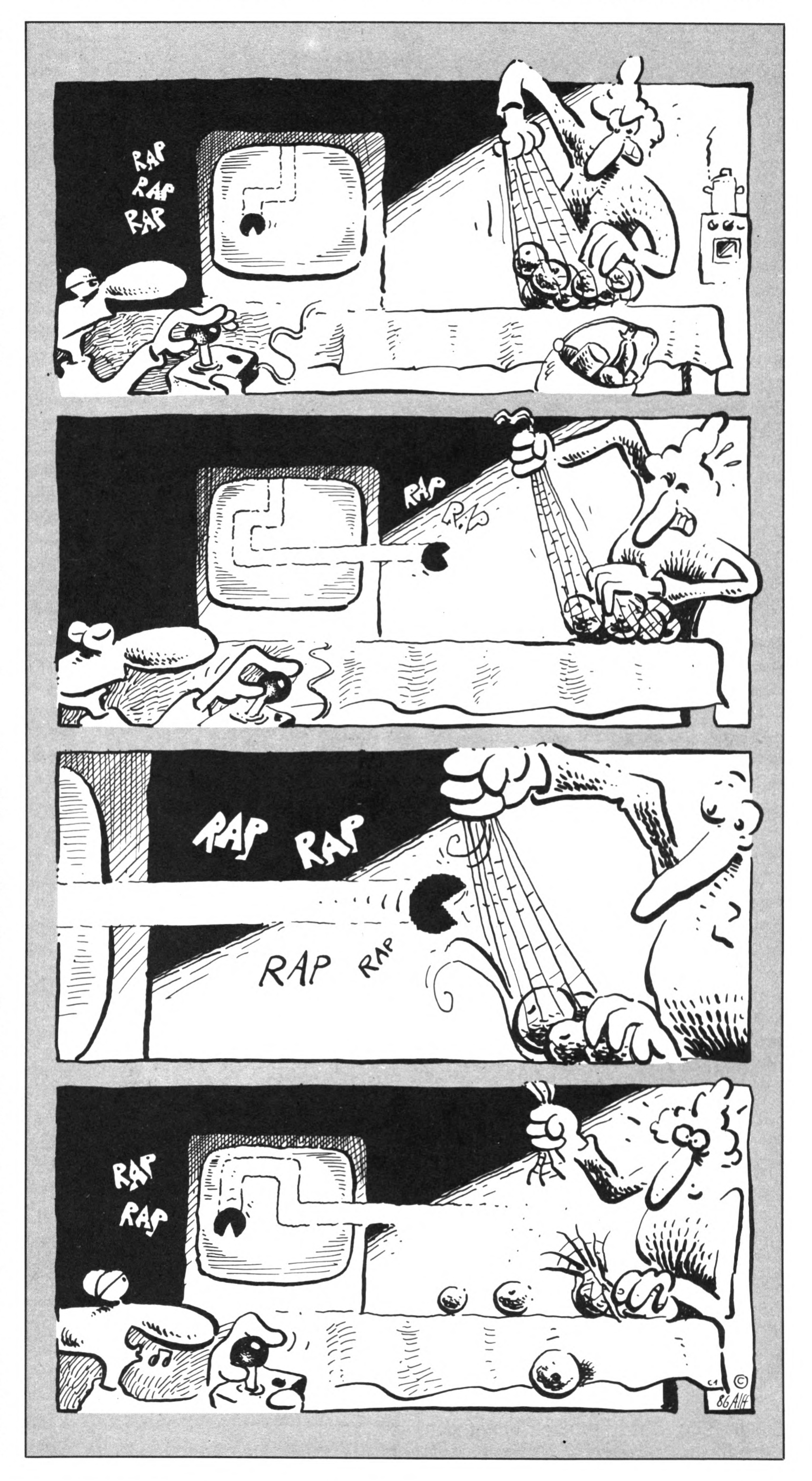
Zahlen können dezimal (23 oder 23D), hexadezimal (0FFH), binär (0101B) und oktal (970 oder 97Q) angegeben werden. Auch hexadezimale Zahlen müssen mit einer Ziffer beginnen, notfalls stellen Sie eine Null voran. Stringkonstanten erkennt der Assembler nur, wenn sie von einfachen Anführungsstrichen (') umgeben sind. Außerhalb von Anführungsstrichen werden alle Kleinbuchstaben in Großbuchstaben umgewandelt. Die Eingabe erfolgt formatfrei. Die einzelnen Teile einer Zeile müssen lediglich durch mindestens ein Leerzeichen oder einen Tabulatorsprung voneinander getrennt sein. Wie Sie Ihr Programm eintippen wollen, bleibt also Ihnen überlassen.

Kommentare zur Dokumentation

Eine vollständige und syntaktisch richtige Quellcode-Zeile könnte für ASM so aussehen: »START: LXI H,$/2 ! DB C9H,00B,'Text'; Kommentar«. Der Assembler läßt sich zum Beispiel mit »ASM file.ABX« aufrufen. Ausnahmsweise bedeuten die drei Zeichen nach dem Punkt im Dateinamen, hier ABX, nicht die Extension des Namens. Quellcode-Dateien müssen nämlich immer mit ASM enden, um mit ASM bearbeitet werden zu können, beispielsweise FILE.ASM, PROGRAM. ASM, CODE.ASM. Das erste Zeichen der Extension beim Aufruf gibt an, auf welchem Laufwerk sich die Quelldatei befindet – also A oder B. Das zweite Zeichen teilt dem Assembler mit, auf welche Diskettenstation er den erzeugten Code schreiben soll. Statt einer Laufwerksbezeichnung können Sie auch Z angeben. Dann erzeugt ASM keinen Code, sondern prüft nur das Assemblerprogramm auf syntaktische Richtigkeit. Das dritte Zeichen der Extension bezeichnet das Laufwerk, auf dem die List-Datei ausgegeben werden soll. Z unterdrückt die Ausgabe, X leitet das Listing auf den Bildschirm um. »ABX« bedeutet dann, daß die Quellcode-Datei auf Laufwerk A steht, der erzeugte Code auf B geschrieben wird und das Listing auf den Bildschirm protokolliert wird. »AAX« speichert auch das erzeugte Programm auf A.

Nach der Übersetzung mit ASM befinden sich zwei weitere Dateien auf der Diskette. Sie haben den gleichen Namen wie die Quelldatei, jedoch eine andere Extension: ».PRN« ist die Listdatei und ».HEX« die Code-Datei.

Diese Code-Datei muß noch von einem anderen Programm zu einem



D =	Data Error Die Zahl ist zu groß für den vorgesehenen Datenbereich.
E =	Espression Error Der Ausdruck kann nicht berechnet werden.
L =	Label Error Fehler bei der Zuweisung eines Labels (Marke).
N =	Not Implemented Diesen Befehl versteht ASM nicht, evtl. aber MAC oder RMAC.
O =	Overflow Der Ausdruck ist zu kompliziert bzw. ein String hat mehr als 64 Zeichen.
P =	Phase Error Das Label hat im Paß 2 einen anderen Wert als im Paß 1.
R =	Register Error Das angegebene Register kann bei diesem Befehl nicht verwendet werden.
S =	Syntax Error Ein falsches Mnemonic wurde gefunden.
U =	Undefined Symbol Ein Symbol ist vor der Verwendung nicht definiert worden.
V =	Value Error Ein Wert ist fehlerhaft.

Bild 9. Fehlermeldungen des ASM

```
                  ;Datei HALLO.ASM
                   ;
0100                      ORG  0100H
                   ;
0100 0E09                 MVI C,9           ;Frage nach Namen
0102 112D01               LXI D,ASK
0105 CD0500               CALL 5
                   ;
0108 0E0A                 MVI C,10          ;Lies den Namen
010A 114501               LXI D,BUF
010D CD0500               CALL 5
                   ;
                   ;      Füge Leerzeichen zwischen
                   ;      "Hallo" und Name ein

0110 214501               LXI H,BUF
0113 3620                 MVI M,20H          ;ASCII Leerzeichen
0115 23                   INX H
0116 5E                   MOV E,M            ;Bestimme Zahl der
                                             ;Zeichen der Antwort
0117 3620                 MVI M,20H
0119 1600                 MVI D,0
011B 23                   INX H
011C 19                   DAD D
011D 3621                 MVI M,'!'
011F 23                   INX H
0120 3624                 MVI M,'S'
                   ;
0122 0E09                 MVI C,9            ;Sage "Hallo"
0124 113E01               LXI D,ANS
0127 CD0500               CALL 5
                   ;
012A C30000               JMP 0000           ;Zurück zum CCP
                   ;
012D 5769652068ASK:        DB 'Wie heißen Sie? S'
013E 0D0A48616CANS:        DB 0DH,0AH,'Hallo'
0145 41          BUF:      DB 65             ;Puffergröße: 65 Zeichen
```

Listing. Ihr CPC lernt Sie kennen

COM-File gemacht werden – von LOAD.COM. Der Aufruf von LOAD erfolgt mit dem Namen der HEX-Datei, aber ohne Extension (Beispiel: »LOAD PROGRAM«).

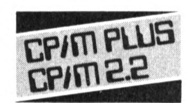
Wenn Sie Ihr Programm also auf den Namen PROGRAM getauft haben, befinden sich jetzt folgende Dateien auf der Diskette: PROGRAM.ASN (der Quellcode), PROGRAM.PRN (die List-Datei), PROGRAM.HEX (die Hex-Datei) und PROGRAM.COM (das ausführbare Programm).

Versuchen Sie doch einmal, das Listing einzugeben, zu speichern und anschließend zu assemblieren. Wenn alles korrekt verläuft, meldet sich das fertige Programm mit der Frage »Wie heißen Sie?«. Nach Eingabe Ihres Namens begrüßt Sie Ihr Computer mit »Hallo« und Ihrem Namen. Dank der Kürze ist das Listing auch für Anfänger noch überschaubar.

Natürlich entdeckt der Assembler auch Fehler in Ihrem Programm. Diese werden in der List-Datei und zusätzlich auf dem Bildschirm mit einem Kennbuchstaben ausgegeben. Das Bild 9 zeigt alle Fehlercodes.

(Martin Kotulla/hg)

Aller Anfang ist leicht



CP/M-Programme schreiben, das heißt in Maschinensprache programmieren. Aber keine Angst, die ersten Versuche sind ganz einfach.

Wenn Sie sich durch die anderen Artikel unseres CP/M-Teils »durchgebissen« haben, dann sollen Sie hier belohnt werden. Wir wollen ein CP/M-Programm schreiben. Zuerst ist es noch ganz einfach. Es wird nur eine Meldung – ein String – auf dem Bildschirm ausgegeben.

Zum Eingeben des Programms nehmen Sie ein beliebiges Textverarbeitungsprogramm – notfalls die unter »Einstieg in CP/M« beschriebene Variante mit »PIP File.ASM=CON:«. ».ASM« deshalb, damit der ASM-Assembler unsere Routine übersetzen kann. Ein Editor (ED) steht auf der Systemdiskette Ihres Schneiders. Wollen Sie mit Wordstar arbeiten (sehr komfortabel für Maschinencode-Eingaben), dann müssen Sie Ihre Eingabe unbedingt im N-Modus (Bearbeiten einer Programmdatei) vornehmen.

In die erste Zeile Ihres Programms gehört ein Kommando, das den Programmzähler auf die Adresse 0100hex, den Beginn der TPA, setzt:

```
ORG 0100H
```

Oder, wenn Sie es übersichtlicher und klarer haben wollen, dürfen Sie ein Label definieren:

```
TPA EQU 0100H
    ORG TPA
```

Damit wir genau wissen, wo sich der Stapelzeiger (SP-Register) befindet, fügen wir einen Ladebefehl ein, der das Register auf den Bereich direkt unterhalb der TPA zeigen läßt. Dort reserviert sich das Betriebssystem einen 128 Byte großen Puffer, den wir jetzt aber noch nicht benötigen:

```
LXI SP,0100H
```

oder

```
SP,TPA
```

Alle CP/M-Funktionen werden über das BDOS (Basic Disk Operating System) aufgerufen. Aber wie macht man das?

1. Man lädt das C-Register mit der Nummer der gewünschten Funktion. Alle BDOS-Funktionen sind durchnummeriert.
2. Wenn 8-Bit-Werte übergeben werden sollen, nutzt man dazu das E-Register, bei 16-Bit-Werten nimmt man das DE-Doppelregister als Speicher für den benötigten Wert.
3. An der Adresse 5 (0005hex) steht ein direkter Sprung ins BDOS. Also ruft man diese Adresse auf. Nach Bearbeitung der BDOS-Funktion fährt der Prozessor unmittelbar nach der Aufrufstelle im Programm fort.

Da wir eine Zeichenkette ausgeben wollen, suchen wir uns eine Funktion heraus, die das macht. Dies ist die BDOS-Funktion 9. Also laden wir das C-Register mit dieser Funktionsnummer (hier der 9):

```
MVI C,9
```

Übersichtlicher und damit auch für andere verständlicher wird es mit einer EQU-Zuweisung:



```
OUTPUTS EQU 9
        MVI C,OUTPUTS
```

Zu lesen ist das Label als »Output String« (Gebe eine Zeichenkette aus).

Da das BDOS wissen muß, an welcher Adresse der String zu finden ist, laden wir das DE-Register mit einem symbolischen Zeiger auf den Text:

```
LXI D,STRING
```

Alle Vorbereitungen sind getroffen und der Aufruf kann an das BDOS abgeschickt werden.

```
BDOS EQU 5
        CALL BDOS
```

Da das Programm schon zu Ende ist, fügen wir gleich noch einen Warmstart ein, so daß der Benutzer wieder im CCP landet:

```
WARM EQU 0
        JMP WARM
```

Doch halt! Wir haben ja noch gar nicht den String angegeben, den wir ausdrucken wollen. Das läßt sich nachholen:

```
STRING DB 'Hallo -- so einfach
ist das!'
```

Ganz Gewissenhafte können jetzt noch »END« dazuschreiben – bei manchen Textsystemen muß man END immer angeben, da sonst ASM »ins Blaue hinein« assembliert. Da hilft dann nur Ausprobieren. Eine Quelle meist »unerklärlicher« Fehler bei der Programmübersetzung durch den ASM ist so heimtückisch, daß fast jeder einmal darüber stolpert: Vergessen Sie keinesfalls bei Wordstar, in der letzten Programmzeile den Zeilensprung (Enter) auszulösen. Andernfalls steigt der ASM ohne jede Fehlermeldung aus!

Tippen Sie das Programm ab und speichern Sie es als »MYFIRST.ASM«. Dem Assembler übergeben Sie es als »ASM MYFIRST.AAA« und dem Lader als »LOAD MYFIRST«.

Schauen Sie sich doch einmal mit »TYPE MYFIRST.PRN« oder »PIP LST: =MYFIRST.PRN« die List-Datei an. Wenn Sie Tippfehler gemacht haben, entdeckt ASM diese und meldet alle Zeichen, die er nicht versteht. Dann müssen Sie nochmals mit dem Editor heran und die beanstandeten Zeilen verbessern. Die Fehlermeldungen finden Sie in Bild 9 des Artikels »Einstieg in CP/M«.

Irgendwann ist ASM dann mit dem Quellcode so zufrieden, daß er keine Fehlermeldungen mehr ausspuckt, und Sie können mit LOAD ein COM-File erzeugen. Wenn der übliche Prompt »A« erscheint, geben Sie »MYFIRST« ein. CP/M lädt Ihr Programm von der Diskette und startet es. Doch was ist das? Er hört gar nicht nach dem kurzen Satz auf, sondern druckt noch einen Haufen Buchstaben aus, die nie und nimmer im Quellcode stehen!

Es kann sogar der Fall eintreten, daß Sie einen Reset auslösen und CP/M neu laden müssen. Bei gründlicher Überlegung können Sie durchaus auf die Fehlerursache kommen. Wir haben der BDOS-Routine gar nicht mitgeteilt, wie lang der auszugebende String ist, beziehungsweise wo er endet!

Wir haben weiter oben erfahren, daß nur das DE-Register laut CP/M-Konvention zur Datenübermittlung benutzt werden darf. Dieses ist aber schon mit

```
;
MYFIRST.ASM  -  Das erste CP/M-Programm
;

; Definitionen ***********************

BDOS     EQU    5          ; Sprung ins BDOS
TPA      EQU    0100H      ; Beginn der TPA in CP/M
WARM     EQU    0          ; Warmstart
OUTPUTS  EQU    9          ; BDOS-Funktion 9: Stringausgabe

; Programm ***************************

         ORG    TPA        ; Programmstart festsetzen

START    LXI    SP,TPA     ;  Stackpointer unter die TPA
         MVI    C,OUTPUTS; BDOS-Funktionsnummer ins C-Register
         LXI    D,STRING  ; Zeiger auf den String
         CALL   BDOS       ; Vom BDOS ausgeben lassen
         JMP    WARM       ; CP/M-Warmstart

STRING   DB     'Hallo - so einfach ist das!',13,10,'$'
         END
```

Bild 1. So einfach ist Ihr erstes CP/M-Programm

```
                ;
                ; MYFIRST.ASM - Das erste CP/M-Programm
                ;

                ; Definitionen ***********************

0005 =          BDOS     EQU  5           ; Sprung ins BDOS
0100 =          TPA      EQU  0100H       ; Beginn der TPA in CP/M
0000 =          WARM     EQU  0           ; Warmstart
0009 =          OUTPUTS  EQU  9           ; BDOS-Funktion 9: String-
                                            ausgabe

                ; Programm ***************************

0100                     ORG  TPA         ; Programmstart festsetzen

0100 310001     START    LXI  SP,TPA      ; Stackpointer unter die TPA
0103 0E09                MVI  C,OUTPUTS   ; BDOS-Funktionsnummer
                                            ins C-Register
0105 110E01              LXI  D,STRING    ; Zeiger auf den String
0108 CD0500              CALL BDOS        ; Vom BDOS ausgeben lassen
010B C30000              JMP  WARM        ; CP/M-Warmstart
010E 48616C6C6FSTRING DB   'Hallo - so einfach ist das!',13,10,'$'
012C                     END
```

Bild 2. So sieht »MYFIRST« nach dem Bearbeiten mit dem Assembler aus

```
:10010000310001 0E09110E01CD0500C30000486148
:100110006C6C6F202D20736F2065696E666163685B
:0C012000206973742064617321 0D0A24AF
:0000000000
```

Bild 3. Das Intel-Hex-Format Ihres Programms ist für Menschen unlesbar



dem Zeiger auf den String belegt. Da die Programmierer bei Digital Research konsequent bleiben wollten, haben sie sich dafür entschieden, das Stringende durch ein Dollarzeichen anzugeben. Bessern wir also die Zeile aus:

```
STRING DB 'Hallo -- so einfach
ist das!',13,10,'$'
```

Wenn Sie das Programm wieder nach der üblichen Prozedur übersetzen und ausführen lassen, erhalten Sie die gewünschte Bildschirmausgabe – und Ihr erstes fehlerfreies CP/M-Programm! Das Bild 1 zeigt den Quellcode, wie er von Ihnen eingegeben werden sollte, Bild 2 die List-Datei und die .HEX-Datei, das sogenannte Intel-Hex-Format. Dieses spezielle Format wird vom Assembler erzeugt und von LOAD in eine COM-Datei ungewandelt.

Bildschirmeffekte, oh, là là!

Kennen Sie zufällig einen Besitzer eines Commodore 128, eines Bondwell-, Sanyo- oder eines anderen CP/M-Computers? Sicher läßt er Sie einmal ein paar Minuten an sein Gerät. Tippen Sie auch dort das Programm ein und lassen Sie es laufen! Es ist schon ein tolles Gefühl, Programme zu schreiben, die auf einer Vielzahl verschiedener Computer ohne Änderungen lauffähig sind.

Übrigens, die Bytes 13 und 10 im String dienen dazu, einen Wagenrücklauf und ein Line Feed (Zeilenvorschub) an den Bildschirm zu senden. Damit hebt sich der ausgedruckte Satz besser von der folgenden »A«-Meldung ab.

Experimentieren Sie ruhig mit dem String. Sie können beispielsweise Steuerzeichen zum Bildschirmlöschen oder zur Farbänderung einfügen. Sofern die Zeichen nicht gerade mit den CP/M-Steuerzeichen kollidieren, ist das erlaubt und kann tolle Bildschirmeffekte hervorrufen. Im Schneider-Handbuch finden Sie eine Tabelle mit allen erlaubten Bildschirm-Steuerzeichen. Wie wäre es etwa mit dieser Zeile?

```
STRING DB 4,1,15,2,28,2,1,24
       DB 'So kann man Texte auch
          darstellen!'
       DB 15,1,13,10,10,10,'$'
```

Die Steuerzeichen sind aber bei jedem Computer anders. Wenn Sie CP/M-Programme mit Bildschirmsteuerung auf andere Geräte übertragen wollen, müssen Sie diese Control-Codes anpassen!

Um wieder in die CP/M-Standard-Darstellungsweise MODE 2 zu gelangen, müssen Sie CTRL-D, CTRL-B und Enter drücken.

Mit diesem Beispiel kennen Sie bereits die erste BDOS-Funktion des CP/M-Betriebssystems. Es bleibt nur noch anzumerken, daß es den BDOS-Routinen erlaubt ist, alle Registerinhalte zu verändern. Wenn Sie also diese Werte weiter benötigen, müssen Sie sie vor dem Aufruf auf den Stack retten und nachher wieder zurückholen. BDOS und BIOS arbeiten im übrigen mit einem eigenen Prozessor-Stack. So müssen Sie nicht mehr Platz für den Stack Ihres Programms reservieren, als es für eigene Zwecke benötigt.

Was machen Sie aber, wenn Sie statt eines ganzen Strings nur ein einziges Zeichen ausgeben wollen? Die Bildung einer Zeichenkette, die aus einem einzelnen Zeichen besteht, ist wohl nicht sehr sinnvoll. Das BDOS stellt stattdessen dafür die Funktion 2 zur Verfügung, die »Konsolenausgabe« genannt wird. Das E-Register wird mit dem ASCII-Code des Zeichens geladen, das Sie ausgeben wollen. Programmieren wir also eine Schleife, die den gesamten ASCII-Zeichensatz von 32 bis 255 druckt! Die üblichen Zeilen, mit denen fast alle CP/M-Programme beginnen, übernehmen wir hier wieder. Im folgenden werden aber nicht mehr alle Definitionen (EQUates) explizit angegeben. Sie werden als bekannt vorausgesetzt und sollten immer von Ihnen in den Quellcode Ihrer Programme eingefügt werden:

```
BDOS EQU 5
TPA EQU 0100H
WARM EQU 0
```

Um eine Schleife mit weniger als 256 Durchgängen zu programmieren, werden Z80-Kenner wahrscheinlich den Befehl DJNZ (Decrement B-Register And Jump If Not Zero) verwenden. Sie laden das B-Register mit der Zahl der Durchläufe und setzen an das Ende der Schleife DJNZ. Dieser Befehl dekrementiert das B-Register und prüft, ob es den Wert Null erreicht hat. Trifft dies nicht zu, ruft der Prozessor den Anfang der Schleife auf.

Schleife für Schleife programmiert

Doch ein 8080-Assembler wie der ASM hat natürlich von den Z80-Mnemonics keine Ahnung, und die Intel-Prozessoren kennen keinen vergleichbaren Befehl. Nehmen wir doch stattdessen den Akku. Der wird sowieso nicht benötigt und kann problemlos vor dem Aufruf der BDOS-Routine gerettet werden.

Versuchen Sie am besten, das folgende Programm selbst nachzuvollziehen:

```
         ORG TPA
         LXI SP,TPA
         MVI A,256-32
         MVI E,32
SCHLEIFE PUSH D
         PUSH PSW
         MVI C,2
         CALL BDOS
         POP PSW
         POP D
         INR E
         DCR A
         JNZ SCHLEIFE
         JMP WARM
         END
```

Maschinensprache-Profis werden bemängeln, daß dieses Programm »entsetzlich umständlich« aufgebaut ist. Schreiben Sie am besten ein gleichwertiges Programm, das nicht den Akkumulator als Schleifenzähler verwendet, sondern nach jeder Zeichenausgabe den Inhalt des E-Registers in den Akku lädt und dort mit 255 vergleicht. Das ist nicht allzu schwer, aber eine gute Übung für jeden Programmierer.

Die Tastatur: Rettung, wenn der Computer nichts weiß

Bisher haben wir zwei Funktionen kennengelernt, mit denen sich Zeichen auf dem Bildschirm ausgeben lassen. Genauso wichtig ist aber auch die Tastaturabfrage. Dazu brauchen Sie aber noch einige Informationen, wie CP/M die Tastatur verwaltet: Während der Schneider im Basic-Betriebssystem einen Zeichenpuffer benutzt, der per Interrupt während des Programmlaufs gedrückte Tasten registriert, ist das in CP/M viel primitiver: Der Tastaturpuffer kann sich nur ein einziges Zeichen merken! Dies hat aber auch den Vorteil, daß ein Löschen des Tastaturpuffers vor wichtigen Eingaben ganz einfach ist.

Die BDOS-Funktion 11 (Konsolenstatus abfragen) stellt fest, ob ein Zeichen im Puffer ist. Dies ist übrigens die erste Funktion, die nicht nur Registerwerte übernimmt, sondern auch Werte an das aufrufende Programm übergibt. Dafür hat Digital Research wieder bestimmte Regeln festgelegt: 8-Bit-Werte meldet das BDOS im Akkumulator, 16-Bit-Werte im HL-Doppelregister.

Die Funktion 11 setzt das A-Register auf Null, wenn kein Zeichen verfügbar ist. Sonst erhält der Akku den Wert 1. Mit diesem Unterprogramm können Sie zum Beispiel sehr einfach die Aufgaben lösen, nach einer Meldung wie »Press any key to continue« oder »Bitte drücken Sie eine Taste« auf die Reaktion des Benutzers zu warten.

Ein kleines Programmbeispiel soll das demonstrieren. Wir verwenden diesmal zwei BDOS-Funktionen, nämlich Nummer 9 (Zeichenkette ausgeben) und Nummer 11 (Konsolenstatus abfragen):

```
        ORG TPA
        LXI SP,TPA
MELDUNG MVI C,9
        LXI D,STRING
        CALL BDOS
WARTEN  MVI C,11
        CALL BDOS
        CMP 0
        JZ WARTEN
        JMP WARM
STRING  DB 'Press any key to'
        DB 'continue.....$'
        END
```

Meistens wollen Sie als Programmierer aber auch wissen, welche Taste sich der Benutzer ausgesucht hat. Dazu haben sich die System-Entwickler bei Digital Research die Funktion 1 (Konsoleneingabe) einfallen lassen. Diese meldet im Akku den Code der gedrückten Taste. Zusätzlich wird das Zeichen auf dem Bildschirm ausgegeben. Die Steuerzeichen CTRL-S und CTRL-P werden beachtet, nicht hingegen CTRL-C, mit dem sich normalerweise ein Warmstart des Betriebssystems auslösen läßt.

Auch diese Funktion wollen wir anhand eines kleinen Programms erläutern. Es fragt zuerst nach einem Tastendruck und gibt dann das Zeichen mit einem Antwort-Text auf dem Bildschirm aus. Wir verwenden wieder nur die als bekannt vorausgesetzten Funktionen:

```
         ORG  TPA
         LXI  SP,TPA
ANFRAGE  MVI  C,9
         LXI  D,TEXT1
         CALL BDOS
TASTATUR MVI  C,1
         CALL BDOS
         PUSH PSW
AUSWERT  MVI  C,9
         LXI  D,TEXT2
         CALL BDOS
         POP  PSW
         MOV  E,A
         MVI  C,2
         CALL BDOS
         JMP  WARM
TEXT1    DB   'Bitte druecken
              Sie eine Taste:
              $'
TEXT2    DB   13,10
         DB   'Sie haben diese
              Taste gedrueckt:
              $'
         END
```

Der Programmteil ANFRAGE gibt den Text »Bitte druecken Sie eine Taste:« aus. TASTATUR fragt die Tastatur ab und rettet das gelesene Zeichen mit PUSH PSW auf den Stack. Der Programmteil AUSWERT schreibt den Text »Sie haben diese Taste gedrueckt:« auf den Bildschirm und gibt zusätzlich das gelesene Zeichen aus.

Problematisch wird es hier, wenn Sie ein Zeichen eingeben, das gleichzeitig ein CP/M-Control-Code ist, beispielsweise CTRL-S oder CTRL-P. Wenn Sie nämlich einen Control-Code an das Schneider-Betriebssystem senden wollen, der zugleich von CP/M benutzt wird, fängt ihn das BDOS ab und läßt ihn gar nicht mehr an den BIOS-Teil durch.

Eine Funktion für zwei Aufgaben

Auch ist es in manchen Fällen nicht erwünscht, daß das von der Tastatur empfangene Zeichen auf dem Bildschirm ausgegeben wird. Dazu wurde die Funktion 6 geschaffen, die direkte Konsolen-Ein- und -Ausgabe. Diese BDOS-Funktion erledigt zwei völlig verschiedene Aufgaben: einerseits die Tastaturabfrage und andererseits die Bildschirmausgabe – beide unter Umgehung des BDOS.

Wenn Sie das E-Register mit einem Wert ungleich 255 (00FFhex) laden, gibt die Funktion das entsprechende Zeichen auf dem Bildschirm aus. So können Sie ohne Probleme den gesamten Schneider-Zeichensatz von 0 bis 254 ausdrucken:

```
         ORG  TPA
         LXI  SP,TPA
         MVI  E,0
SCHLEIFE PUSH D
         MVI  E,1
         MVI  C,6
         CALL BDOS
         POP  D
         PUSH D
         MVI  C,6
         CALL BDOS
         POP  D
         INR  E
         MOV  A,E
         CPI  254
         JNZ  SCHLEIFE
         JMP  WARM
         END
```

Diese Routine ist recht einfach zu verstehen, wenn man weiß, daß der Schneider auch die Control-Codes von 0 bis 31 auf dem Bildschirm als Zeichen ausgeben kann. Allerdings muß ihnen ein »CHR$(1)« vorangehen, ähnlich »PRINT CHR$(1);CHR$(13)« in Basic.

Vor der Schleife wird das E-Register, das wie üblich das zu druckende Zeichen enthält, auf Null gesetzt. In der Schleife rettet der Computer zuerst das DE-Register und gibt CHR$(1) direkt über das BIOS aus. Der Zeichencode wird wieder ins E-Register geladen und gleich darauf vor dem BDOS gerettet. Die BDOS-Funktion 6 gibt nun das eigentliche Zeichen aus. Vergleichsbefehle sind nur mit dem Akkumulator zulässig, also lädt das Programm den ausgegebenen Buchstaben in den Akku und vergleicht ihn dort mit 254. Ist dieser Wert erreicht, verläßt das Programm die Schleife und kehrt mit einem Warmstart auf die CCP-Kommandoebene zurück.

Die zweite Aufgabe der Funktion 6 ist die ungepufferte Tastaturabfrage, bei der das gelesene Zeichen nicht auf dem Bildschirm ausgegeben wird. Das BDOS erkennt Ihren Wunsch, die Tastatur abzufragen und nicht etwa ein Zeichen auszugeben, daran, daß das E-Register den Wert FFhex (255) enthält. Als Resultat liefert die Routine den ASCII-Wert der gedrückten Taste. Wenn keine Taste im Abfragezeitraum registriert wurde, meldet die Funktion den Wert Null. Auch das ist ein Unterschied zur Funktion 1 (Konsoleneingabe). Die direkte Konsoleneingabe wartet nicht, bis ein Zeichen zur Verfügung steht. Am deutlichsten wird der Unterschied wohl, wenn wir das Programm, mit dem die Konsoleneingabe-Funktion erläutert wurde, auf die Funktion 6 umschreiben:

```
         ORG  TPA
         LXI  SP,TPA
ANFRAGE  MVI  C,9
         LXI  D,TEXT1
         CALL BDOS
TASTATUR MVI  C,6
         MVI  E,255
         CALL BDOS
         CPI  0
         JZ   TASTATUR
         PUSH PSW
AUSWERT  MVI  C,9
         LXI  D,TEXT2
         CALL BDOS
         POP  PSW
         MOV  E,A
         MVI  C,2
         CALL BDOS
         JMP  WARM
TEXT1    DB   'Bitte druecken
              Sie eine Taste:
              $'
TEXT2    DB   13,10
         DB   'Sie haben diese
              Taste gedrueckt:
              $'
         END
```

Die Änderungen beschränken sich auf den Teil TASTATUR. C=6 ruft die direkte Konsolen-Ein- und -Ausgabe auf, durch E=255 selektiert das Programm die Tastaturabfrage. Meldet das BDOS den Tastenwert Null im Akku, wurde keine Taste gedrückt und das Programm kehrt zurück in die Schleife. Andernfalls gibt der Computer die entsprechende Meldung aus. Wenn Sie Lust haben, können Sie das Programm so umschreiben, daß die Bildschirm-

ausgabe nicht über die BDOS-Funktion 2 läuft, sondern über die Funktion 6 – genauso wie in unserem zweiten Zeichensatz-Beispielprogramm mit CHR$(1). Dann können Sie wirklich jede Taste drücken, ohne befürchten zu müssen, daß irgendeine Systemebene das Zeichen als Steuerzeichen mißversteht und nicht weiterreicht.

Doch in vielen Fällen wollen Sie nicht ein einzelnes Zeichen von der Tastatur abrufen, sondern einen ganzen String.

Eine ganze Zeichenkette

Dazu benötigen Sie einen Befehl ähnlich INPUT in Basic. Für viele Maschinencode-Programmierer ist es ein Graus, alle Editierfunktionen und das Ein- und Ausschalten des Cursors zu installieren. CP/M bietet eine sehr komfortable Lösung: eine eigene BDOS-Funktion. Sie ist sozusagen das Gegenstück zur Funktion 9 (Zeichenkette ausgeben) und heißt entsprechend »Zeichenkette einlesen«.

Ihre Funktionsnummer ist 10. Der Aufruf ist etwas komplizierter als bei den bisherigen Funktionen. Das DE-Register wird mit einem Zeiger auf einen Speicherbereich geladen, der als Eingabepuffer verwendet wird. Dorthin schreibt später das BDOS alle eingegebenen Zeichen, so daß Sie sie im Programm lesen und auswerten können.

Das erste Byte des Pufferspeichers sollten Sie mit der maximal erlaubten Länge der Eingabezeile laden. Die Werte von 1 bis 255 sind möglich. Wenn der Benutzer mehr als die erlaubte Zeichenzahl einzutippen versucht, kehrt das BDOS in das aufrufende Programm zurück, ohne auf das Drücken der ENTER-Taste zu warten. Etwas ähnliches erleben Sie, wenn Sie versuchen, im CCP eine Zeile von mehr als 128 Zeichen einzugeben.

Nach dem Aufruf enthält das zweite Byte im Puffer die Zahl der wirklich eingegebenen Zeichen. In den übrigen Bytes des Eingabepuffers finden Sie die Werte der eingetippten Symbole. An Editierfunktionen ist alles vorhanden, was auch der CCP-Editor bietet: DEL, CTRL-H, CTRL-U, CTRL-M, CTRL-J, CTRL-R, CTRL-X, CTRL-E und leider auch CTRL-C. Wird CTRL-C als erstes Zeichen in der Zeile eingetippt, bricht der Computer das laufende Programm ab und springt zum CCP zurück. Schreiben wir ein Programm, das eine Eingabe von zehn Zeichen vom Benutzer fordert und diesen Text in der Folge 20mal auf dem Bildschirm ausdruckt:

```
                ORG    TPA
                LXI    SP,TPA
EINGABE         MVI    C,10
                LXI    D,PUFFER
                CALL   BDOS
```

Neben der üblichen Einleitung lädt das Programm das C-Register mit der Funktionsnummer 10 und das DE-Register mit einem Zeiger auf den – noch zu definierenden – Pufferspeicher. »CALL BDOS« leitet die Befehlsausführung ein.

```
AUSGABE     MVI    A,20
SCHLEIFE    MVI    C,9
            LXI    D,PUFFER+2
            PUSH   PSW
            CALL   BDOS
            POP    PSW
            DCR    A
            JNZ    SCHLEIFE
            JMP    WARM
```

Als Schleifenzähler verwenden wir hier das A-Register. Die Funktion 9 gibt einen String auf dem Bildschirm aus. Das DE-Register wird mit PUFFER+2 geladen, weil die ersten beiden Bytes ja Informationen über die Länge der Zeile geben – und die werden bei der Ausgabe nicht benötigt. Ist der Akku auf Null heruntergezählt, führt das Betriebssystem einen Warmstart durch.

Das wichtigste ist aber hier der Puffer, der durch geschickten Aufbau eine Menge Programmieraufwand ersparen kann:

```
PUFFER      DB 10
            DB 0,32,32,32,32,32
            DB 32,32,32,32,32
            DB 13,10,'$'
            END
```

Der Pufferaufbau läßt sich so zerlegen:

1. Byte: Maximale Länge der Benutzereingabe.
2. Byte: Beliebig, das BDOS trägt nach der Befehlsausführung hier die tatsächliche Länge der Eingabezeile ein.
3. bis 12. Byte: Pufferspeicher, der hier mit Leerzeichen vorbelegt ist.
13. Byte: Wagenrücklauf für die Bildschirmausgabe.
14. Byte: Zeilenvorschub bei der BDOS-Funktion 9.
15. Byte: Stringende-Kenner bei der Zeichenkette-Ausgabe (Funktion 9).

Ebenfalls zeichenorientiert: Drucker, Lochstreifenleser und -stanzer

Damit kennen Sie alle Befehle zur Bedienung von Tastatur und Bildschirm. Es gibt noch drei andere zeichenorientierte Peripheriegeräte: den Lochstreifenleser, den Lochstreifenstanzer und – nicht zu vergessen – den Drucker. Zeichenorientiert heißen diese Geräte deshalb, weil man von ihnen immer nur ein Zeichen lesen, beziehungsweise an sie ein einzelnes Zeichen übergeben kann. Die anderen Peripheriegeräte wie Diskette oder Festplatte können auch ganze Bytefolgen lesen und schreiben. Aufgrund ihrer Konstruktion sind sie außerdem in der Lage, eine spezielle Spur oder einen Sektor direkt anzusprechen.

Wohl kein Heimcomputer-Besitzer wird ein Lochstreifen-Gerät sein Eigen nennen. Deshalb der Vollständigkeit halber nur kurz die beiden zugehörigen BDOS-Routinen:

Funktion 3: Lochstreifen lesen. Der Akku liefert das gelesene Zeichen.

Funktion 4: Lochstreifen stanzen. Im E-Register muß der Code des auszugebenden Zeichens stehen.

Interessant sind die beiden Funktionen nur noch, weil sie manchmal zur Ansteuerung einer seriellen Schnittstelle oder eines Kassettenrecorders »mißbraucht« werden. Beide lassen sich aber beim Schneider nur mit Schwierigkeiten anpassen. Die serielle Schnittstelle von Schneider arbeitet nämlich nicht mit den System-Routinen im BIOS-ROM zusammen und der Kassettenrecorder benötigt zur Datenübergabe immer einen 2 KByte großen Pufferspeicher. Bei einer TPA von rund 39 KByte unter CP/M 2.2 (464 und 664 ohne Speichererweiterung) kann man sich dieses »Speicherplatzopfer« nicht mehr leisten.

Viel wichtiger ist die Druckeransteuerung unter CP/M. Die Funktion 5 (Zeichen auf dem Drucker ausgeben) übernimmt entsprechend dem Standard im E-Register das auszugebende Zeichen. Ist der Drucker nicht empfangsbereit, wartet die BDOS-Funktion, bis sie das Zeichen übergeben kann.

Wer sich das ROM-Listing des BIOS-Teils im Bereich C4A1hex bis C4A8hex und die zugehörigen Unterprogramme anschaut, kann erkennen, wie der Hersteller des Schneider-Computers das Verhalten bei nicht empfangsbereitem Drucker abfängt – oder auch nicht abfängt. Je nach Gerätetyp kann der Computer aussteigen, in einer Endlosschleife verharren oder sonstige unerwünschte Resultate zeigen. Beim CPC 464 verbleibt das Programm in einer Endlosschleife, die nur durch CTRL-C oder einen Reset mit CTRL-SHIFT-ESCAPE abgebrochen werden kann.

Als Beispiel schreiben wir ein Programm, das einen Satz auf dem Drucker ausgibt. Dieser Text soll im Quellcode festlegbar sein. Der Programmkopf ist wieder CP/M-typisch:

```
            ORG    TPA
            LXI    SP,TPA
```

Das HL-Doppelregister dient uns als Zeiger auf den String.

```
            LXI    H,STRING
```

Die Programmschleife wird beendet, wenn der Computer ein Dollar-Zeichen gelesen hat. Diesen Stringbegrenzer kennen wir ja schon aus der BDOS-Funktion 9 und wollen ihn hier übernehmen:

```
SCHLEIFE MOV  A,M
         CPI  '$'
         JZ   ENDE
         MOV  E,A
         MVI  C,5
         INX  H
         PUSH H
         CALL BDOS
         POP  H
         JMP  SCHLEIFE
ENDE     JMP  WARM
```

Zuerst lädt der Prozessor einen Buchstaben in den Akku und vergleicht ihn mit dem Dollarsymbol. Fällt der Vergleich positiv aus, so geht der Computer zur Marke ENDE und löst dort einen Warmstart aus. Ist hingegen das Stringende noch nicht erreicht, lädt das Programm das gelesene Zeichen ins E-Register – eine Vorbereitung für den BDOS-Ansprung. »MVI C,5« spezifiziert die gewünschte Funktion, und CALL BDOS ruft das BDOS direkt auf. Da dieses alle Register verändern kann, das HL-Register aber nicht zerstört werden darf, sichern wir HL mit »PUSH H« auf dem Stack und holen es später mit »POP H« wieder zurück. »INX H« vor dem PUSH-Befehl weist den Computer an, das HL-Register auf den nächsten Eintrag im String zeigen zu lassen.

Als String können Sie irgendeinen Text einsetzen, der Ihnen gefällt, zum Beispiel:

```
STRING   DB 'So spricht CP/M den
            Drucker an'
         DB 13,10,10,'$'
```

Wo ist was?

Das IOBYTE, manchmal auch I/O-Byte oder Input/Output-Byte geschrieben, merkt sich im Betriebssystem die Zuordnung der logischen zu den physikalischen Geräten. Diese Unterscheidung wurde schon beim Programm STAT.COM (Einstieg in CP/M) erklärt. Das IOBYTE liegt im Speicher an der Adresse 0003hex. Wenn Sie es direkt verändern, können Sie auf den Einsatz von STAT.COM verzichten. Allerdings ist das IOBYTE nicht allzu wichtig. Wer ändert schließlich ständig seine Hardware? Zudem sollten Sie von Manipulationen des IOBYTES absehen, wenn Ihre Programme auch unter CP/M-3.0 (CP/M plus) funktionieren sollen. Diese Variante von CP/M verwendet das IOBYTE gar nicht mehr, sondern besitzt viel leistungsfähigere Alternativen. Deshalb muß das IOBYTE dem Standard entsprechen.

Zu jedem der vier logischen Geräte lassen sich je vier physikalische Peripheriebausteine zuordnen. Bild 5 in dem Artikel »Einstieg in CP/M« zeigt die symbolischen Gerätenamen. Im IOBYTE sind die Zuweisungen binär codiert. Für jedes logische Gerät sind zwei Bit reserviert. Mit diesen zwei Bit lassen sich vier Werte darstellen – eben die vier physikalischen Geräte.

Das wichtigste Peripheriegerät unter CP/M

Welche Bits welches Gerät betreffen, das zeigt Ihnen die folgende Aufstellung:
- Bits 0 und 1: die Konsole CON:
- Bits 2 und 3: der Lochstreifenleser RDR:
- Bits 4 und 5: der Lochstreifenstanzer PUN:
- Bits 6 und 7: Der Drucker LST:

Auslesen läßt sich das IOBYTE mit der BDOS-Funktion 7 (IOBYTE). Der Inhalt des Bytes wird im A-Register übermittelt. Gesetzt wird das IOBYTE mit der Funktion 8 (IOBYTE setzen). Im E-Register übergeben Sie dazu den neuen Inhalt.

Sie kennen jetzt alle Befehle, die mit den vier zeichenorientierten Peripheriegeräten (CON:, RER:, PUN:, und LST:) zu tun haben. Die zweite Hauptgruppe der BDOS-Funktion tut eigentlich das, was man vom Namen »BDOS« (Basic Disk Operating System) erwartet: Sie ermöglicht die Arbeit mit Diskettendateien.

Bevor wir uns an die Diskette selbst wagen, benötigen wir einige grundlegende Informationen. Der Computer muß wissen, welche Datei angesprochen werden soll. Dazu übermitteln wir ihm die Adresse eines »Dateischreibers«. Dieser wird FCB genannt, ausgeschrieben: File Control Block. Auf Deutsch etwa »Dateikontrollblock«. Ehe wir irgendeine Datei ansprechen können, müssen wir einen FCB zusammenstellen. FCBs haben ein Standardformat, das immer gleich aufgebaut ist:

0. Byte: Disketten-Laufwerk
1. bis 8. Byte: Dateiname
9. bis 11. Byte: Extension

Die übrigen der 36 Byte im FCB nutzt CP/M intern zur Verwaltung der Datei. Einige Speicherstellen des FCB dürfen Sie auch selbst verwenden.

Die FCB-Blöcke im Speicher korrespondieren mit den Einträgen im Inhaltsverzeichnis. Wird eine Datei geöffnet, kopiert CP/M den Directory-Eintrag von der Diskette in den FCB. Die diversen Lese- und Schreiboperationen verändern den FCB automatisch. Beim Schließen einer Datei schreibt der Computer dann anhand des FCB einen neuen Eintrag ins Disketteninhaltsverzeichnis.

FCBs müssen immer exakt nach Vorschrift aufgebaut sein. So muß etwa der Dateiname genau acht Stellen umfassen. Ist er kürzer, ist er mit Leerzeichen (Spaces) aufzufüllen. Das gilt analog für die Extension. Der Punkt zwischen Dateiname und Extension wird nicht angegeben. Das Diskettenlaufwerk ist im Byte 0 des FCB codiert: Eine 1 wählt das Laufwerk A, eine 2 das Laufwerk B und so weiter. Geben Sie eine Null an, holt sich CP/M die Diskette vom Standardlaufwerk, das im CCP-Prompt (A> oder B>) angegeben ist.

Eine besondere Schwäche von CP/M ist es, daß das Betriebssystem die Diskettendateien immer nur in 128 Byte großen »Happen« lesen kann. Andere Betriebssysteme wie beispielsweise MS-DOS für IBM-Computer sind da viel flexibler. Mit etwas Programmieraufwand kann man aber mit dieser Einschränkung ganz gut zurechtkommen. Die Bezeichnung »Sektor« für die 128-Byte-Einheiten ist etwas unglücklich gewählt. Intern verarbeitet der Schneider unter Amsdos und im CP/M-BDOS nämlich Sektoren von 512 Byte Länge. Es obliegt dem BIOS, diese in vier Teile 128 Byte zu zerlegen und beim Schreiben wieder zusammenzufügen. Eingebürgert hat sich deshalb der Begriff »logischer Sektor« für das, was das BDOS unter Sektoren versteht, im Gegensatz zu den »physikalischen Sektoren« des BIOS. Logische Sektoren lassen sich auch als »Records« bezeichnen.

Wohin mit dem Sektor?

Der Programmierer muß dem Betriebssystem mitteilen, wohin es den gelesenen – logischen – Sektor schreiben soll. Diese Adresse heißt in der CP/M-Terminologie DMA (Direct Memory Access).

Versuchen wir, ein Programm zu schreiben, das eine bestehende Datei öffnet, einen Datensatz liest, auf dem Bildschirm anzeigt und die Datei wieder schließt. Dazu sollten Sie natürlich erst einmal eine Datei erzeugen. Tun Sie das mit »PIP A:DATEI.FIL=CON:«. Sie können einen beliebigen Satz eingeben. Am Schluß verwenden Sie bitte das Dollarzeichen, ENTER, CTRL-J und CTRL-Z.

Verzichten Sie aber unbedingt darauf, das Dollarsymbol im übrigen Satz zu benutzen – die BDOS-Funktion 9, mit der wir den Satz auf dem Bildschirm ausgeben wollen, stellt sich sonst stur

und druckt einfach nur den ersten Teil des Satzes aus. Eine grundlegende Änderung sollten wir für die zukünftigen Programme noch machen. Wir verlegen den Prozessor-Stack in einen anderen Speicherabschnitt, weil wir den Bereich zwischen 0080hex und 00FFhex für andere Zwecke brauchen:

```
ORG  TPA
LXI  SP,STACK
```

Da wir genau wissen wollen, wohin CP/M den Datensatz schreibt, setzen wir mit der BDOS-Funktion 26 (Datenpuffer festlegen) die DMA-Adresse. Das DE-Register ist mit einem Zeiger auf den Speicherbereich zu laden:

```
LXI  D,BUFFER
MVI  C,26
CALL BDOS
```

Alle folgenden Diskettenzugriffe bis zum nächsten Warmstart verwenden nun den – noch zu definierenden – Puffer »BUFFER« als Zwischenspeicher.

Ein Programm im Detail

Das waren alles noch Vorarbeiten. Doch jetzt wird es spannend. Wir öffnen unsere erste Diskettendatei:

```
LXI  D,FCB
MVI  C,15
CALL BDOS
```

Natürlich haben wir noch keinen FCB festgelegt – das machen wir am Ende des Programms. Sie brauchen sich nur zu merken, daß die BDOS-Funktion 15 eine bereits bestehende Datei öffnet, wenn das DE-Register auf den FCB zeigt.

Die nächste BDOS-Funktion, die wir verwenden wollen, liest einen Datensatz von der Diskette ein. Sie besitzt die Funktionsnummer 20 und erwartet wieder im DE-Register einen Zeiger auf den Dateikontrollblock:

```
LXI  D,FCB
MVI  C,20
CALL BDOS
```

Jetzt steht der Satz im Speicher und muß nur noch auf den Bildschirm gebracht werden. Da ja am Ende des Satzes das Dollarzeichen als Stringbegrenzer steht (Sie haben es doch hoffentlich nicht vergessen?), können wir ohne weiteres die BDOS-Funktion 9 (Zeichenkette ausgeben) aufrufen:

```
LXI  D,BUFFER
MVI  C,9
CALL BDOS
```

Da Sie ein »gewissenhafter« CP/M-Programmierer werden sollen, erscheint es ratsam, daß Sie sich gleich zu Anfang angewöhnen, alle Dateien – auch die nur gelesenen – wieder zu schließen. Die vorgesehene BDOS-Funktion »Datei schließen« trägt die Nummer 16:

```
LXI  D,FCB
MVI  C,16
CALL BDOS
```

Einen Warmstart schicken wir noch nach:

```
JMP  WARM
```

Die eigentliche Arbeit beginnt allerdings erst jetzt. Sie müssen noch Vereinbarungen über alle Pufferspeicher treffen. Fangen wir mit dem einfachsten an: dem Stack. 20 Byte entsprechen 10 Stapeleinträgen und reichen im Normalfall aus. Damit lassen sich alle vier Doppelregister sichern und bis zu sechsfach verschachtelte Unterprogramme aufrufen:

```
         DS   20
STACK    EQU  $
```

Wem die Reihenfolge der Definitionen (erst DS, dann die Marke STACK) nicht einleuchtend ist, der muß daran denken, daß der Stack nach unten wächst und das SP-Register immer mit dem oberen Ende des Stapelspeichers geladen wird.

Auch der DMA-Puffer ist schnell erzeugt:

```
BUFFER   DS   128
```

Verzwickter wird die Lage beim File-Control-Block. Hier geben wir zuerst die Laufwerknummer an, gefolgt vom Dateinamen und der Extension. Da der FCB immer 36 Byte lang ist, müssen wir genügend Platz reservieren. Wichtig ist es auch noch, die Bytes 12 und 32 im FCB beim Öffnen der Datei auf Null zu setzen. Fügen wir doch ganz einfach für alle Bytes zwischen 12 und 36 Nullen ein:

```
FCB      DB   1
         DB   'DATEI'
         DB   32,32,32,
         DB   'FIL'
         DB   0,0,0,0,0,0
         DB   0,0,0,0,0,0
         DB   0,0,0,0,0,0
         DB   0,0,0,0,0,0,0
```

Noch ein »END« daruntergesetzt, und das Programm kann assembliert werden.

Mehr Power

Das Byte 32 im FCB-Block hat eine besondere Bedeutung. Dort speichert das BDOS die Nummer des als nächstes zu bearbeitenden Records. Diese Eigenschaft können Sie ausnutzen, indem Sie dort die Nummer des von Ihnen gewünschten Datensatzes angeben. Sogar schon beim Öffnen der Datei kann der als erstes zu lesende Record vorgegeben werden.

Ändern Sie also die letzte DB-Zeile Ihres Programmes so ab:

```
DB   0,0,3,0,0,0,0
```

Wenn das File DATEI.FIL lang genug ist, können Sie so den dritten Datensatz der Datei lesen.

Ohne weiteres ist es möglich, in ein und derselben Datei gleichzeitig zu lesen und zu schreiben. Dies ist im Vergleich zu Amsdos ein großer Fortschritt. Verwenden Sie wieder den Quellcode Ihres Programms und bringen Sie die letzte DB-Zeile wieder in den alten Zustand mit lauter Nullen. Die folgenden Zeilen sind von Ihnen vor der Stringausgabe (Funktion 9) einzusetzen.

```
LXI  H,BUFFER+3
MVI  M,254
```

Durch diese beiden Befehle ersetzt der Computer das dritte Byte im gelesenen Datensatz durch das Zeichen 254 – sinnvoll oder nicht, hier kommt es lediglich darauf an, das Wissen zu vermitteln.

Zum Schreiben von Records gibt es die BDOS-Funktion 21. Sie erwartet im DE-Register einen Zeiger auf den FCB-Block und schreibt den Record, der sich im aktuellen DMA-Puffer befindet, auf die Diskette:

```
LXI  D,FCB
MVI  C,21
CALL BDOS
```

Den Rest der alten Datei können Sie wieder übernehmen. Austesten läßt sich das COM-Programm mit Hilfe eines kleinen Basic-Programms. Drücken Sie also nach der Assemblierung CTRL-SHIFT-ESCAPE und geben Sie dieses Programm ein:

```
10 OPENOUT "DATEI.FIL"
20 FOR i=1 TO 50
30 PRINT #9, "Datensatz";
   STR$(i);"$";
40 NEXT i
50 CLOSEOUT
```

Nach »RUN« gelangen Sie mit dem RSX-Befehl »|CPM« nach CP/M zurück. Drücken Sie dort TYPE DATEI.FIL, so gibt der CCP den Dateiinhalt aus. Bearbeitet wird der Dateiinhalt, indem Sie das COM-File, das Sie gerade eben übersetzt haben, starten. Dieses gibt den Datensatz mit dem eingefügten neuen Zeichen auf dem Bildschirm aus und schreibt ihn in die Datei. Sobald sich das Betriebssystem mit A> oder B> meldet, können Sie sich mit TYPE DATEI.FIL die Datei erneut anzeigen lassen – der Unterschied dürfte deutlich sein. Doch etwas Unerwartetes geschieht: Geändert wurde nicht der erste Datensatz, sondern der erste Record wurde zusammen mit der Änderung in den zweiten Record geschrieben. Das ist eine Sonderleistung von CP/M, da es automatisch nach jedem Schreib- oder Lesezugriff den internen Zähler erhöht. Normalerweise ist das Einlesen der Records der Reihe nach sehr sinnvoll. Das selbsttätige Hoch-

zählen des Zeigers führt in den meisten Fällen zu kürzeren Programmen.

Aber anstatt mit dem Record-Zeiger und sequentiellen Dateien zu hantieren, ist es meistens einfacher, eine Direktzugriff-Datei zu programmieren (siehe auch 2. Schneider-Sonderheft 1/86).

Die Datei DATEI.FIL zeigt übrigens, daß es ohne weiteres möglich ist, ASCII-Textdateien zwischen Basic und CP/M hin- und herzuschicken. Bei Binärdateien und Basic-Programmen ist das schwieriger, da diese einen sogenannten File-Header (Kopfsatz) besitzen, der Nicht-ASCII-Werte enthalten kann.

Wenn Sie ein Programm schreiben, das endlose Datensätze in einem File unterbringt, ist irgendwann einmal der Zeitpunkt gekommen, daß Ihre Datei größer als 16 KByte wird. Das mag CP/M aber nicht besonders gern, weil es so große Dateien mit einem einzigen Eintrag im Diskettendirectory nicht verwalten kann. Die Lösung ist aber ganz einfach. CP/M »spendiert« dem Programm einen weiteren Eintrag im Inhaltsverzeichnis. Bei DIR taucht er aber nicht auf, da er besonders gekennzeichnet ist. Diese zusätzlichen Einträge heißen »Extents«. Dateien können so nicht nur 16 KByte, sondern auch 32, 64, 128 KByte oder mehr sein. Die Grenze setzen die Diskettenkapazität und die Aufnahmefähigkeit des Directories. Schneider-typisch ist, daß das Directory nur 64 Einträge verwalten kann. Also gehen Sie möglichst sparsam mit diesen um. Besonders Besitzer der ersten Version der Vortex-Floppy können von der Kapazität des Inhaltsverzeichnisses ein Lied singen. Auf einer Diskette bringt man dort über 700 KByte Daten unter – das Directory faßt aber trotzdem nur 64 Einträge. VDOS 2.0, das neue DOS der Vortex-Station, kann nun aber 128 Einträge verwalten. Und das reicht meist aus.

Die Extents werden vom BDOS automatisch durchnummeriert, damit sie korrekt bearbeitet werden können. Diese Information legt das BDOS auch im FCB-Block ab. Das Byte 12 ist dafür reserviert. Bei entsprechendem Bedarf können Sie darauf in Ihren Programmen zugreifen. Zum Beispiel sind beim Lesen von Dateien im Byte 32, das die Record-Nummer angibt, nur Werte zwischen 0 und 127 gestattet. Wollen Sie auf dahinter liegende Einträge zurückgreifen, müssen Sie beim Öffnen der Datei im Byte 12 die Nummer des Extents angeben und im Byte 32 dann die Sektornummer, abhängig vom Beginn des Extents.

Fehlererkennung

Fehler sind menschlich – sie können entweder Ihnen als Programmierer unterlaufen oder dem Benutzer Ihres Produkts. Egal, woher der Fehler stammt, man muß erst einmal wissen, daß er überhaupt aufgetaucht ist und dann Abhilfe schaffen.

Fehlerquellen sind zum Beispiel die Angabe falscher Dateinamen, das Lesen über das Dateiende hinaus, das Schreiben von Dateien auf volle Disketten oder in volle Directories – der Möglichkeiten gibt es viele. Sofern das BDOS, das eine Diskettenroutine entsprechend Ihrem Auftrag ausführt, etwas von einem Fehler bemerkt, können Sie sicher sein, daß es Ihnen davon eine Mitteilung schickt. Der »Bote« ist der Akkumulator. Wenn dieser nach dem Aufruf von BDOS-Routinen bestimmte Werte aufweist, ist davon auszugehen, daß etwas schiefgelaufen ist.

Die Routinen 20 (Record lesen) und 21 (Record schreiben) liefern im Akkumulator den Wert 0, wenn alles geklappt hat. Sie können Ihr Programm um eine Fehlerbehandlung erweitern. Fügen Sie nach dem Lesen des Records (LXI D,FCB ! MVI C,20 ! CALL BDOS) folgende Zeilen ein:

```
        CPI   0
        JNZ   ERROR
```

Durch den Vergleich mit Null stellt der Computer fest, ob ein Fehler aufgetreten ist. In diesem Fall springt er zur Marke ERROR. Dort muß eine Meldung an den Benutzer ausgegeben werden. Bei komplexeren Programmen ist es empfehlenswert, gewisse Vorkehrungen wie beispielsweise das Schließen der benutzten Dateien, das Löschen des Bildschirms und ähnliche Dinge aufzurufen. Begnügen wir uns mit einem lakonischen Fehlertext:

```
ERROR    LXI  D,STRING
         MVI  C,9
         CALL BDOS
         JMP  WARM
STRING   DB   'Bei READ  ist ein
              Fehler aufgetreten
              !$'
```

Testen können Sie die Funktionsfähigkeit dieser Fehlerbehandlungs-Routine, indem Sie ganz einfach die Datei DATEI.FIL mit REName umbenennen und damit für das Programm sozusagen »unsichtbar« machen.

Die anderen bisher erwähnten Diskettenfunktionen melden Fehler ebenfalls im Akku, allerdings in anderer Form. Wenn der Akkuinhalt 255 (00FFhex) beträgt, ist irgend etwas beim Diskettenzugriff schief gelaufen. Sie könnten zum Beispiel direkt nach dem Funktionsaufruf zum Öffnen der Datei einen Error-Check einbauen:

```
ERRCHECK CPI   255
         JZ    OPENERROR
```

Und am Ende des Programms:

```
OPENERROR LXI  D,MESSAGE
          MVI  C,9
          CALL BDOS
          JMP  WARM
MESSAGE   DB   'Bei OPEN ist
               ein Fehler
               aufgetreten.'
```

(Martin Kotulla/hg)









Programmieren unter CP/M



Im zweiten Teil unseres CP/M-Programmierkurses lernen Sie, wie man Dateien erzeugen und Sie dann auf der Diskette speichern kann. Nach diesem Artikel werden Sie alle BDOS- und alle BIOS-Funktionen Ihres Schneider-Computers perfekt beherrschen.

Im ersten Teil unseres CP/M-Kurses (»Aller Anfang ist leicht«) haben wir nur bestehende Dateien gelesen und verändert. Nun wollen wir einmal selbst eine Datei erzeugen und auf der Diskette speichern. CP/M bietet die BDOS-Funktion 22 (Datei erzeugen) an. Sie verlangt im DE-Register einen Zeiger auf den benutzten FCB-Block. Der FCB-Block muß wie üblich aufgebaut werden. Doch es gibt leider noch einige Fallstricke. Die Funktion prüft zum Beispiel nicht, ob bereits eine Datei mit einem identischen Namen existiert, sondern legt ohne Rücksicht auf Verluste einen neuen Verzeichniseintrag im Directory an. Ein Resultat kann sein, daß auf der Diskette ein Chaos herrscht. Abhilfe schafft ein Befehl zum Löschen von eventuell bestehenden Dateien dieses Namens, der vorangeschickt werden sollte. Eleganter ist der Weg, eine bestehende Datei zum .BAK-File umzubenennen und die neue Datei unter dem geplanten Namen abzulegen. Programmtechnisch ist der Aufwand für diese Lösung größer. Schließlich müssen Sie noch prüfen, ob eine .BAK-Datei dieses Namens bereits existiert.

Bevor wir also eine neue Datei erzeugen können, müssen wir uns erst mit den Befehlen zum Löschen und Umbenennen von Dateien auseinandersetzen. Zur Erklärung des Löschbefehls wollen wir ein Äquivalent für den ERA-Befehl schreiben.

Bisher sind Sie gewohnt, sich selbst den FCB-Block zusammenzubasteln. Das erfordert nur wenig Mühe, wenn sich der Dateiname nicht ändert, sondern fest im Programmcode verankert ist. Stellen Sie sich aber vor, Sie müßten eine Benutzereingabe auswerten und in einen korrekten Dateinamen umsetzen. Einige Seiten Quellprogramm kommen dabei schnell zusammen. Da erinnern Sie sich daran, bei verschiedenen CP/M-Programmen nach dem Namen des Hauptprogramms weitere Dateinamen angegeben zu haben. Zum Beispiel bei WordStar »WS BRIEF.BRF«. Diese Eingabe nennt sich »CCP-Kommandozeile« und muß vom Betriebssystem irgendwo abgespeichert werden. Und tatsächlich, der Pufferspeicher liegt zwischen den Adressen 0080 hex und 00FF hex. Damit verstehen Sie auch, warum es sinnvoll ist, den Z80-Stack nicht gerade in diesen Bereich zu legen.

Damit der Aufbau der Kommandozeile erkennbar wird, sollten Sie sich eine Diskette mit dem Systemdebugger DDT heraussuchen und diesen mit »DDT FILE.COM« laden. Auch wenn Sie gar kein Programm »FILE.COM« besitzen, hat das keine unangenehmen Auswirkungen. Der DDT meldet sich dann zwar mit einem »?«, unternimmt aber keine weitergehenden Aktionen.

Mit dem D-Befehl (Dump) listet er den Speicher auf. Geben Sie die Start- und Endadresse an, zeigt der Debugger den Inhalt des angesprochenen Speicherbereiches in Hexadezimalzahlen und ASCII-Zeichen an. So listet »DC000,FFFF« das komplette Schneider-BIOS-ROM. Anhalten läßt sich die Ausgabe mit CTRL-S, abbrechen mit jeder beliebigen anderen Taste. CTRL-P schaltet wahlweise den Drucker zu, der auf Wunsch alle Ausgaben mitprotokolliert.

Lassen Sie sich die zweite Hälfte der Zero-Page mit D0080,00FFF ausgeben. Sie muß in etwa so aussehen, wie sie im Bild 1 wiedergegeben ist. Sofern Sie keine unnötigen Leerzeichen beim Aufruf von DDT eingegeben haben, finden Sie an der Adresse 0080 hex den Wert 09 hex. Welche Bedeutung das Byte hat, können Sie herausfinden, wenn Sie die Länge Ihrer Eingabe nachzählen. »DDT FILE.COM« besitzt 12 Stellen. Wenn Sie von 12 die drei Buchstaben »DDT« abziehen, erhalten Sie die Zahl 9. Das Byte an der Adresse 0080 hex gibt also die Länge der Eingabezeile abzüglich der Namenslänge des aufgerufenen Programms an. Das Leerzeichen zwischen dem Programmnamen und den angegebenen Parametern wird mitgezählt. Der Aufruf »PIP LST:=CON:« speichert dann also eine 10 (A hex) an der Adresse 0080 hex.

In den Bytes zwischen 0081 und 00FF hex findet sich dann die gesamte Eingabezeile wieder. Wurde eine Kommandozeile mit weniger als 128 Buchstaben eingegeben, ist der Inhalt der restlichen Bytes unbestimmt. Das Betriebssystem siedelt nämlich auch den Sektorpuffer zur Kommunikation mit der Diskettenstation in diesem Bereich an. Wenn Sie die Auflistung in Bild 1 betrachten, finden Sie beispielsweise einen Auszug aus dem Directory der im letzten Kapitel bearbeiteten Diskette. Die Programmnamen READCMD.PRN, READCMD.HEX und READCMD.COM zeugen davon.

Mit dieser Kommandozeile lassen sich an Programme beliebige Parameter übermitteln. Als Demonstration schreiben wir eine Routine, die die Minimalfunktionen einer Schreibmaschine erfüllt. Es soll eine Zeile aus dem CCP-Kommandopuffer übernehmen und auf dem Drucker ausgeben. Das Programm ist aber aus zwei Gründen nur sehr beschränkt zur Textbearbeitung geeignet. Erstens wandelt es alle Kleinbuchstaben in ihre großen Äquivalente um, und zweitens müßten Sie das Programm für jede Druckzeile neu von der Diskette laden.

Beginnen wir mit dem üblichen Programmkopf:

```
TPA     EQU     0100H  } Diese Adressen
WARM    EQU     0      } gelten wieder für
BDOS    EQU     5      } alle Programme
ORG     TPA
LXI     SP,STACK
```

Als nächstes laden wir den Akkumulator mit der Länge der Eingabezeile. So läßt er sich als Schleifenzähler verwenden:

```
LDA     0080H
```

Das HL-Register zeigt auf den Beginn des Kommandopuffers:

```
LXI  H,0081H+1
```

Der Abstand 0081hex+1 ist erforderlich, weil wir ja schon gesehen haben, daß in 0081hex immer ein Leerzeichen steht. Das wollen wir aber nicht drucken.

In einer Programmschleife wird nun der ganze Text auf dem Drucker ausgegeben. Dazu stellt uns das BDOS die –

bereits bekannte – Funktion 5 (Zeichen auf dem Drucker ausgeben) zur Verfügung:

```
SCHLEIFE  MOV   E,M
          INX   H
          DCR   A
          JZ    ENDE
          MVI   C,5
          PUSH  PSW
          PUSH  H
          CALL  BDOS
          POP   H
          POP   PSW
          JMP   SCHLEIFE
```

Wenn der Akku auf Null dekrementiert wurde, ruft der Prozessor die Befehle ab der Marke ENDE (JZ ENDE) auf. Sonst erhöht das Programm den Zeiger im HL-Register, sichert die Register A und HL auf dem Stack, gibt das Zeichen über das BDOS aus, stellt die Registerinhalte wieder her und kehrt zurück in die Programmschleife.

Am Programmende ist es empfehlenswert, einen Wagenrücklauf und Zeilenvorschub an den Drucker zu schicken:

```
ENDE      MVI   E,13
          MVI   C,5
          CALL  BDOS
LINEFEED  MVI   E,10
          MVI   C,5
          CALL  BDOS
          JMP   WARM
STACK     EQU   $
          END
```

Mit dem Auslesen des Kommandopuffers sind wir aber unserem Ziel, uns die Konstruktion von FCBs zu ersparen, keinen Schritt näher gekommen. Aber CP/M bietet einen bemerkenswerten Komfort. Bevor es das von der Diskette geladene COM-File aufruft, stellt es fest, ob vom Benutzer zusätzliche Dateinamen angegeben wurden und versucht, einen FCB-Block zu erstellen.

Laden Sie wieder den Systemdebugger mit DDT FILE.COM und listen Sie mit D005C,0080 einen anderen Teil der Zero-Page. Bild 2 zeigt, wie es dort aussieht. Wenn Sie den Hexdump genau betrachten, werden Sie feststellen, daß er einen vollständigen Dateikontrollblock darstellt.

So können wir sehr leicht unser Ziel verwirklichen, einen ERA-Ersatz zu schreiben. Als zusätzliche Leistung soll das Programm den von CP/M aufgebauten FCB ausdrucken und eine Sicherheitsabfrage machen:

```
      ORG   TPA
FCB   EQU   005CH
START LXI   SP,STACK
```

Zuerst soll das Programm den Dateinamen so zeigen, wie er von CP/M im Standard FCB gespeichert wird. Wie Sie aus unserem letzten Kursteil bereits wissen, beginnt der Dateiname an der Adresse FCB+1. An der Adresse FCB steht das gewählte Diskettenlaufwerk.

Lassen wir zuerst das Laufwerk anzeigen. Es ist binär codiert, also A=1, B=2 und so weiter. Durch Addition von 64 erhalten Sie den Kennbuchstaben (1+64=65=»A«, 2+64=66 =»B«). Geben Sie keine Laufwerksbezeichnung an, so setzt CP/M den Wert Null ein, der dem aktuellen Bezugslaufwerk entspricht. In diesem Fall zeigt das Programm 0+64=64, also einen »Klammeraffen« (@) an:

```
LAUFWERK  LDA   FCB
          ADI   64
          MOV   E,A
```

Die normale Konsolenausgabe (Funktion 2) kann ohne Einschränkungen verwendet werden:

```
          MVI   C,2
          CALL  BDOS
```

Jetzt fehlt noch der Doppelpunkt nach der Laufwerksbezeichnung:

```
          MVI   E,':'
          MVI   C,2
          CALL  BDOS
```

Und nun kommt der Dateiname dran. Er besteht aus acht Buchstaben, zuzüglich drei Buchstaben für die Extension. Der Einfachheit halber fassen wir beide Teile zusammen und geben sie auf einmal aus. Damit sind elf Buchstaben zu drucken:

```
DATEINAME LXI   H,FCB+1
          MVI   A,11
SCHLEIFE  MOV   E,M
          PUSH  PSW
          PUSH  H
          MVI   C,2
          CALL  BDOS
          POP   H
          POP   PSW
          INX   H
          DCR   A
          JNZ   SCHLEIFE
```

Damit beenden wir das Programm und setzen die folgenden Zeilen an den Schluß:

```
          JMP   WARM
          DS    20
STACK     EQU   $
          END
```

Wenn Sie das Programm jetzt assemblieren, können Sie das entstehende COM-File aufrufen und sehen, was CP/M aus Ihrer Eingabe macht. Angenommen, Sie haben das File »ERASE.COM« genannt, sind in untenstehender Tabelle einige Beispiele zu finden, was auf dem Drucker ausgegeben wird.

Soweit funktioniert also alles einwandfrei. Es fehlt nur noch die Sicherheitsabfrage und der eigentliche Löschvorgang. Entfernen Sie nun alle Befehle nach der Zeile JNZ SCHLEIFE und setzen Sie dafür folgende Kommandos ein:

```
ABFRAGE   LXI   D,SICHER$
          MVI   C,9
          CALL  BDOS
```

Diese drei Befehle geben den Text aus, der unter SICHER$ zu finden sein wird. Nun müssen wir noch die Tastaturabfrage einbauen. Die Funktion 1 (Konsoleneingabe) ist hier am geeignetsten:

```
          MVI   C,1
          CALL  BDOS
```

Diese Routine liefert im Akku den ASCII-Wert der gedrückten Taste. Schauen wir uns einmal die Ergebnisse an. Sie müssen nur noch den String SICHER$ definieren und die alten Zeilen in folgende Reihenfolge umstellen:

```
          JMP   WARM
SICHER$   DB    '- Sind Sie
                 sicher? $'
          DS    20
STACK     EQU   $
          END
```

Wenn Sie alles richtig übernommen haben, läuft das Programm zur vollsten Zufriedenheit – und wir können endlich die Dateien, die gelöscht werden sollen, wirklich aus dem Directory entfernen:

```
          CPI   'J'
          JNZ   WARM
ERASE     MVI   C,19
          LXI   D,FCB
          CALL  BDOS
          JMP   WARM
```

Die Befehle müssen noch vor dem String SICHER$ eingesetzt werden. Bild 3 zeigt das vollständige Listing. Sie sehen, nach all dem – leider notwendigen – »Vorgeplänkel« eine wirklich kompakte Lösung mit nur sechs Programmschritten.

Noch eine Kurzinformation zur BDOS-Funktion 19: Der Name lautet »Datei löschen«, die Funktion erwartet im DE-Register einen Zeiger auf den FCB-Block.

In der jetzigen Version kennt ERASE bei der Frage »Sind Sie sicher?« nur »J« als Großbuchstaben für »Ja«. Bei allen anderen gedrückten Tasten löst das Programm einen Warmstart aus, ohne die Datei zu löschen. ERASE steht

Eingabe		Ausgabe
A)ERASE FILE.COM	-	@:FILE COM
A)ERASE A:FILE.C?M	-	A:FILE C?M
A)ERASE *.COM	-	@:????????COM
A)ERASE DATEINAM.*	-	@:DATEINAM???
A)ERASE *.*	-	@:???????????
A)ERASE B:*.H?X	-	B:????????H?X
A)ERASE B:FILENAME.BAK	-	B:FILENAMEBAK

Das macht CP/M aus Ihrem COM-File

natürlich Ihren Erweiterungen offen. Zum Beispiel kann man das Programm so ausbauen, daß auch der Kleinbuchstabe »j« bei der Sicherheitsabfrage akzeptiert wird.

Wie reagiert aber CP/M, wenn der Benutzer einen zweiten Dateinamen in der CCP-Aufrufzeile angibt? Ganz einfach: Es legt auch einen zweiten FCB an. Mehr als zwei FCBs verkraftet das Betriebssystem beim Aufruf von Programmen allerdings nicht.

Der zweite FCB wird direkt nach dem ersten an der Adresse 006C hex abgelegt, er beginnt also genau 16 Byte nach dem ersten. Für die Funktion »Datei umbenennen« ist dies sehr nützlich, denn sie erwartet die beiden Dateinamen in genau diesem Format. Einzige Einschränkung ist, daß das Byte 0 des zweiten FCB auf Null gesetzt sein muß. Es gibt ja das Laufwerk des neuen Dateinamens an. Unterschiedliche Laufwerke beim Umbenennen von Dateien sind natürlich Unsinn. Fassen wir also zusammen:

```
005C hex Byte 0  Laufwerk
005D hex Byte 1  Alter Dateiname
006C hex Byte 16 Neues Laufwerk,
                 immer Null
006D hex Byte 17 Neuer Dateiname
```

Im Gegensatz zum REN-Befehl verlangt also die BDOS-Funktion, daß der alte Dateiname zuerst steht. Das Programm kann recht kurz bleiben:

```
FCB       EQU  005CH
FCB2      EQU  FCB+16
          ORG  TPA
          LXI  SP,STACK
NULLSETZ  LXI  H,FCB2
          MVI  M,0
RENAME    MVI  C,23
          LXI  D,FCB
          CALL BDOS
          JMP  WARM
          DS   20
STACK     EQU  $
          END
```

Die verwendete BDOS-Funktion hat die Nummer 23 und den Namen »Datei umbenennen«. Sie erwartet – wie alle Dateifunktionen – im DE-Register einen Zeiger auf den FCB-Block. Dieser FCB-Block enthält ausnahmsweise zwei Dateinamen. Den Aufbau haben wir gerade besprochen.

Wenn Sie das Programm RENAME.COM nennen, können Sie es statt des REN-Befehls verwenden:

»REN PROG.NEU=PROG.ALT« entspricht »RENAME PROG.ALT PROG.NEU«.

Für diesen Einzelfall ist die Lage des zweiten FCB mitten im ersten äußerst praktisch, doch in den meisten Fällen muß man ihn kopieren. Wenn Sie nämlich die erste Datei mit dem FCB an der Adresse 005C hex öffnen, überschreibt CP/M alle Daten im zweiten FCB. Der FCB hat ja bekanntlich eine Länge von 32 bis 36 Byte und nicht 16 Byte. Z80-Programmierern fällt hier sofort der Blockkopierbefehl LDIR ein. Doch ein 8080-Assembler kann diesen nicht verstehen. Da eine Simulation durch eine Programmschleife recht aufwendig ist, wollen wir uns anschauen, wie wir den ASM dazu bringen können, auch Z80-Befehle zu übersetzen.

Sehr hilfreich ist bei dieser Arbeit eine Vergleichstabelle der Z80- und 8080-Befehle sowie eine Tabelle der Z80-spezifischen Befehle (siehe den Artikel »Z80/8080 im Vergleich«).

Ohne Probleme lassen sich Befehle übersetzen, die keine Parameter besitzen wie LDIR, OTDR, EXX, NEG und RETI. Fügen Sie einfach die Opcodes als DB-Konstanten ein und schon haben Sie den 8080-Assembler. Statt LDIR schreiben Sie nun DB 0EDH, 0B0H, statt EXX jetzt DB 0D9H. Mit einem kleinen Trick lassen sich auch die Indexregister IX und IY ansprechen. Die meisten der Indexregister-Befehle entstehen nämlich aus normalen HL-Register-Kommandos, die das vorangehende Byte mit DD hex (für IX) oder FD hex (für IY) verändert. Eine kleine Tabelle solcher Umcodierungen finden Sie hier:

Z80:	8080:
ADD IX,BC	DB DDH und DAD B
ADD IY,BC	DB FDH und DAD B
LD (adr),IX	DB DDH und SHLD adr
LD SP,IY	DB FDH und SPHL
LD IY,wort	DB FDH und LXI H,wort

Etwas verzwickter wird es bei denjenigen Indexbefehlen, die einen Verschiebewert benötigen. Dieser muß hinter dem Befehl angefügt werden:

Z80	8080
ADD A,(IX+ distanz)	DB 0DDH und ADD M und DB Distanz
CP (IY+ distanz)	DB 0FDH und CMP M und DB Distanz
LD L,(IX+ distanz)	DB 0DDH und MOV L,M und DB Distanz

Alle übrigen Befehle müßten Sie wohl oder übel direkt von Hand ausrechnen und als Konstanten in den Quelltext einfügen.

Nach diesem Ausflug in die Tiefen des Mikroprozessors kehren wir wieder auf die Benutzer- und Programmiererebene von CP/M zurück. Wir wissen zwar jetzt, wie sich Dateien löschen und umbenennen lassen, haben aber immer noch keine Datei erzeugt.

Benutzen wir zuerst unsere Kenntnisse über das Löschen von Files, da alte Dateien unbedingt von der Diskette entfernt werden müssen, bevor die BDOS-Funktion »Datei erzeugen« (Nummer 22) aufgerufen werden darf. Das Programm, das wir schreiben wollen, soll eine Datei erzeugen und einen Satz in diese Datei schreiben.

Wir verwenden dazu den CCP-Kommandopuffer:

```
FCB   EQU  005CH
      ORG  TPA
      LXI  SP,STACK
```

Da wir uns die Arbeit nicht allzu schwer machen wollen – Programmierer arbeiten schließlich nicht, sie lassen den Computer für sich arbeiten – benutzen wir wieder den von CP/M angebotenen FCB-Block. Das erfordert allerdings Disziplin bei der Bedienung, weil sich im Dateinamen bei dieser BDOS-Funktion keine unerlaubten Zeichen befinden dürfen (auch kein ? oder *).

Vorher sollten wir aber noch den DMA-Puffer auf einen neuen Bereich festlegen. CP/M benutzt den Bereich 0080 hex bis 00FF hex. Dort liegt auch die Kommandozeile, die wir später in die Datei schreiben wollen. Sicherheitshalber setzen wir den DMA-Puffer auf diese Adresse:

```
      LXI  D,0080H
      MVI  C,26
      CALL BDOS
```

Die BDOS-Funktion 26 legt den DMA-Speicher neu fest. Löschen lassen sich Dateien mit der BDOS-Funktion 19:

```
      LXI  D,FCB
      MVI  C,19
      CALL BDOS
```

Nach diesen vielen Vorarbeiten ist es soweit. Die Funktion 22 erzeugt eine Diskettendatei, die durch den FCB beschrieben wird. Auf diesen muß das DE-Register zeigen:

```
      LXI  D,FCB
      MVI  C,22
      CALL BDOS
```

Jetzt steht die Datei auf der Diskette. Sie muß nur noch mit Daten gefüllt werden. Nichts einfacher als das:

```
      LXI  D,FCB
      MVI  C,21
      CALL BDOS
```

Diese BDOS-Funktion 21 überträgt 128 Byte, beginnend an der Adresse 0080 hex, auf der Diskette. Auch das Schließen der Datei dürfen wir nicht vergessen:

```
      LXI  D,FCB
      MVI  C,16
      CALL BDOS
```

Am Programmende darf natürlich der Warmstart nicht fehlen, ebensowenig die Stackdefinition:

```
      JMP  WARM
      DS   20
STACK EQU  $
      END
```

Aufrufen sollten Sie das Programm, nachdem Sie es als CREATE.COM gespeichert haben, zusammen mit dem Namen der zu erzeugenden Datei und darauffolgend beliebigen Buchstaben. Geben Sie solange Buchstaben ein, wie der CCP sie annimmt. Das Pro-



gramm bildet dann die neue Datei und schreibt Ihre Eingabezeile in die Datei. Mit TYPE filename.ext können Sie sich ansehen, ob alles nach Wunsch geklappt hat.

Direkter Zugriff auf Daten

Was in Amsdos nur mit einigen Mühen und langen Maschinencode-Routinen simuliert werden kann, bietet CP/M serienmäßig: die Direktzugriffdateien. Im Gegensatz zu sequentiellen Dateien, bei denen Datensätze nur der Reihe nach gelesen werden können, lassen sich in Direktzugriffdateien einzelne Records direkt ansprechen. Damit haben Sie die Möglichkeit, effiziente Dateiverwaltungen mit relativen oder indexsequentiellen Zugriffsverfahren zu entwickeln.

Diese Dateien unterscheiden sich in CP/M nicht allzu sehr von anderen Dateien. Sie werden über genau dieselben BDOS-Funktionen erzeugt (22), geöffnet (15), geschlossen (16), gelöscht (19) oder umbenannt (23) wie alle anderen Files auch. Lediglich für das Schreiben und Lesen sind eigene Funktionen vorhanden.

Für den direkten Zugriff muß der FCB, der bisher nur 32 Byte umfaßte, auf 36 Byte erweitert werden. In den Byte 33 und 34 speichert der CP/M-Computer die Nummer der zu bearbeitenden Aufzeichnung. Wenn Sie einen bestimmten Datensatz bearbeiten wollen, müssen Sie in diesen beiden Bytes die Satznummer eintragen, das Lowbyte zuerst.

Beim Öffnen der Datei ist unbedingt darauf zu achten, die Aufzeichnungsnummer im Byte 32 des FCB-Blocks auf Null zu setzen. Von Manipulationen dieses Werts, die bei der sequentiellen Verarbeitung durchaus zu empfehlen sind, ist beim Direktzugriff abzuraten.

Im Gegensatz zur sequentiellen Dateiverwaltung zählt das BDOS bei direktem Datenzugriff nicht automatisch den internen Zeiger auf den nächsten Datensatz fort. Das muß Ihr Programm machen. Recht nützlich ist aber, daß man jederzeit zwischen sequentieller und direkter Dateiverwaltung hin- und herwechseln darf. Der Zähler im Byte 32 wird nämlich mitverwaltet. Es ist damit aber keineswegs so, daß jedem logischen Datensatz auch ein physikalischer Datensatz entspricht. Denn leere Datensätze werden von CP/M nicht mit abgespeichert, was eine Menge Speicherplatz auf der Diskette spart, den Wechsel zwischen den Betriebsarten aber erschwert.

Die BDOS-Funktion 34 (Aufzeichnung schreiben) verlangt, daß im DE-Register ein Zeiger auf den 36-Byte-FCB zu finden ist, und daß die Bytes 33 und 34 die Satznummer enthalten. Die Funktion schreibt dann eine 128 Byte lange Aufzeichnung in die Datei.

Die BDOS-Funktion 33 (Aufzeichnung lesen) arbeitet ähnlich, liefert aber 128 Byte aus der Datei in den aktuellen DMA-Puffer.

Die BDOS-Funktion 40 (Initialisierung schreiben) schreibt eine aus Nullbytes bestehende Aufzeichnung in die Datei. Damit lassen sich Aufzeichnungen löschen oder als ungültig markieren.

Ist der Wechsel vom direkten auf den sequentiellen Zugriff relativ einfach, stehen Sie bei der umgekehrten Richtung vor großen Problemen. Schließlich schreibt das Betriebssystem bei sequentiellem Aufruf nur das Byte 32 fort, nicht aber die Bytes 33 und 34. Abhilfe schafft da eine BDOS-Funktion, die den Inhalt von Byte 32 in die Numerierung für den unmittelbaren Zugriff umrechnet. Sie besitzt die Nummer 36 und benötigt im DE-Register einen Zeiger auf den File-Control-Block. Nach Bearbeitung der Routine steht die berechnete Nummer in den Bytes 33 und 34 des FCB. Die so errechnete Aufzeichnung liegt allerdings genau eine Aufzeichnung hinter der letzten sequentiellen, weil CP/M ja den Zugriffszeiger fortgeschrieben hat.

Die Erkennung von Fehlern bei Direktzugriff ist – eigentlich ungewöhnlich für das spartanisch ausgestattete CP/M 2.2 – recht reichhaltig. Es gibt insgesamt sechs Fehlercodes, die bei den BDOS-Funktionen 33, 34 und 40 auftreten können. Steht im Akkumulator nach deren Aufruf der Wert Null, konnte die Aktion ordnungsgemäß durchgeführt werden. Die Werte 1 bis 6 sind für die folgenden Fehlermeldungen reserviert:

Code 1 = Leseversuch in noch nicht geschriebener Aufzeichnung
Code 2 = Nicht benutzt
Code 3 = Verzeichniseintrag kann nicht geschlossen werden
Code 4 = Leseversuch in Aufzeichnung ohne Verzeichniseintrag
Code 5 = Verzeichnisüberlauf
Code 6 = Diskette voll

Zwei BDOS-Funktionen für den Diskettenbetrieb haben wir Ihnen bisher »verheimlicht«. Sie sind nicht unbedingt lebenswichtig und werden auch nur selten benutzt. Die Funktion 17 sucht auf der Diskette nach einem Directory-Eintrag, der zu dem Dateinamen im FCB paßt:

```
MVI    C,17
LXI    D,FCB
CALL   BDOS
```

Wenn das BDOS solch einen Eintrag findet, enthält der DMA-Puffer den 128-Byte-Sektor aus dem Directory. Das A-Register zeigt an, welcher der vier 32-Byte-Einträge, die jetzt im Puffer stehen, gemeint ist:

A = 0: Erster Eintrag, Adresse DMA
A = 1: Zweiter Eintrag, Adresse DMA+32
A = 2: Dritter Eintrag, Adresse DMA+64
A = 3: Vierter Eintrag, Adresse DMA+96

Die Funktion 18 sucht – nach dem Aufruf der BDOS-Funktion 17 – nach dem nächsten passenden Eintrag. Das DE-Register wird dazu nicht benötigt:

```
MVI    C,18
CALL   BDOS
```

Die Ergebnisse, die wir dabei erhalten, entsprechen der vorhergehenden Funktion.

Noch mehr nützliche Dinge

Neben der zeichenorientierten Kommunikation und der Diskettenverwaltung bietet das BDOS von CP/M 2.2 noch eine Reihe weiterer brauchbarer Funktionen.

So waren Sie zum Beispiel bisher gewohnt, einen Warmstart des Computers mit JMP 0 auszulösen. Alternativ dazu können Sie die BDOS-Funktion 0 (Warmstart auslösen) aufrufen. In den meisten Fällen gibt es zwischen beiden Methoden keinen Unterschied im Resultat. Manche Programme verbieten aber den Reset-Einsprung an der Adresse 0000 hex. Während hier der direkte Sprungbefehl versagt, kann mit der BDOS-Funktion noch ein korrekter Programmabschluß erreicht werden:

```
MVI   C,0
CALL  BDOS
```

CP/M 2.2 verlangt vom Benutzer nach einem Diskettenwechsel einen Warmstart mit CTRL-C. Der Sinn dieser Maßnahme ist es, zu verhindern, daß Teile von Dateien auf verschiedene Disketten geschrieben werden und dadurch das Diskettendirectory zerstört wird. Ein solcher Warmstart bricht aber jedes laufende Programm ab. Die BDOS-Funktion 13 setzt hingegen das Diskettensystem zurück. Das bedeutet, daß das Laufwerk A im User-Bereich 0 selektiert, ein eventuell bestehender Schreibschutz entfernt und der DMA-Puffer wieder auf die Adresse 0080 hex gelegt wird. Wenn Sie diese BDOS-Funktion in Ihrem Programm aufrufen, kann vom Benutzer der Software ohne Bedenken die Diskette gewechselt werden. Dennoch sollten Sie darauf achten, vor dem Diskettenwechsel alle

Dateien auf der alten Diskette zu schließen. Dazu dient die Funktion 13:

```
        MVI   C,13
        CALL  BDOS
```

Oft ist es allerdings nicht sinnvoll, das gesamte Diskettensystem neu zu initialisieren. Die BDOS-Funktion 37 erlaubt es, einzelne Disketten zurückzusetzen. Dazu muß das DE-Register eine Binärzahl enthalten, in der die anzusprechenden Laufwerke codiert sind. Das CP/M-System kann insgesamt 16 Laufwerke verwalten. Da eignet sich ein 16-Bit-Register wie DE prächtig zur Codierung. Für jedes Laufwerk, das initialisiert werden soll, wird ein einzelnes Bit auf eins gesetzt, ausgehend vom niederwertigsten Bit für das Laufwerk A.

Hier finden Sie die Werte für die beiden Laufwerke A und B:

A zurücksetzen:
00000000$00000001bin = 01hex
B zurücksetzen:
00000000$00000010bin = 02hex
A und B:
00000000$00000011bin = 03hex

(Das Dollarzeichen darf bei ASM.COM zur Gliederung von Binärzahlen verwendet werden).
Der Aufruf der BDOS-Funktion erfolgt mit:

```
        MVI   C,37
        LXI   D,Code
        CALL  BDOS
```

Nach diesem BDOS-Call müssen die zurückgesetzten Laufwerke wieder neu aktiviert werden. Dies geschieht, sobald der Benutzer entweder ausdrücklich ein Laufwerk wählt (»A:« oder »B:«) oder dem Dateinamen einen Laufwerksnamen voranstellt (»A:FILE.COM« TYPE B:TEXT«). Sie können diesen Neuzugriff deutlich bemerken, denn die Diskettenstation läuft hörbar lauter und länger an.

Umgekehrt zur Funktion 37 können Sie mit einer speziellen Funktion feststellen, welche Laufwerke aktiv sind. Die Funktionsnummer lautet 24. Nach dem BDOS-Aufruf steht im HL-Register eine Zahl, in der die aktiven Laufwerke genauso codiert sind wie bei der Funktion 37. Zur Erläuterung dieser Funktion schreiben wir eine kurze Routine, die alle aktiven Laufwerke ausgeben soll. Der Programmkopf entspricht unserem Standard:

```
        ORG   TPA
        LXI   SP,STACK
```

Zuerst soll der Text »Aktive Laufwerke:« ausgegeben werden. Verwendung findet die altbekannte DBOS-Funktion 9 (Zeichenkette ausgeben):

```
DRUCKE  LXI   D,STRING
        MVI   C,9
        CALL  BDOS
```

Nun holen wir uns mit der BDOS-Funktion 24 (Aktive Laufwerke feststellen) den Binärcode ins HL-Register:

```
HOLE    MVI   C,24
        CALL  BDOS
```

Da wir den Wert noch länger benötigen, er aber von den BDOS-Routinen überschrieben werden kann, sichern wir ihn auf dem Stack:

```
        PUSH H
```

Von diesem Wert interessieren uns für zwei Laufwerke nur die beiden niederwertigen Bits im L-Register. Durch den Z80-Bit-Testbefehl BIT läßt sich feststellen, welchen Wert ein Einzelbit eines Registers besitzt. Abhängig vom Ergebnis setzt oder löscht die Z80-CPU das Zero-Flag. Die Maschinencodes des dem 8080-Prozessor unbekannten Befehls simulieren wir durch den DB-Pseudobefehl: BIT 0,L entspricht DB 0CBH, 045H und BIT 1,L ist DB 0CBH,04DH.

```
TEST$A  DB    0CBH,045H
        JZ    TEST$B
```

Sobald das Laufwerk A aktiv ist, wird der Sprung nicht mehr ausgeführt, da das Bit 0 dann 1 ist. Wir wollen dann eine Meldung ausgeben:

```
A$AKTIV LXI   D,AKTIV1$
        MVI   C,9
        CALL  BDOS
        JMP   TEST$B
AKTIV1$ DB    24,'A:',24,32,'$'
```

Das Steuerzeichen 24 in der DB-Zeile schaltet die Inversedarstellung ein und wieder aus. Die Abfrage des Laufwerks B erfolgt ähnlich:

```
TEXT$B  POP   H
        DB    0CBH,04DH
        JZ    WARM
B$AKTIV LXI   D,AKTIV2$
        MVI   C,9
        CALL  BDOS
        JMP   WARM
AKTIV2$ DB    24,'B:',24,32,'$'
```

Den String, der zum Programmstart ausgegeben wird, dürfen wir genauso wenig vergessen wie den Systemstack:

```
STRING      DB     12,'Aktive
                   Laufwerke: $'
            DS     20
STACK       EQU    $
            END
```

Das Programm wird allerdings nicht zweimal hintereinander dieselben Resultate liefern, denn es beendet sich selbst durch einen Warmstart, der alle Laufwerke bis auf A zurücksetzt:

```
A>DIR B:
A>ACTIVE
   Aktive Laufwerke: A: B:
A>ACTIVE
   Aktive Laufwerke: A:
A>B:
B>A:ACTIVE
   Aktive Laufwerke: A: B:
```

Die Funktion 14 (Bezugslaufwerk bestimmen) legt fest, auf welchem Diskettenlaufwerk gearbeitet wird, wenn nicht ausdrücklich der Dateiname FCB ein bestimmtes Laufwerk vorgibt:

```
        MVI   C,14
        MVI   E,Laufwerk
        CALL  BDOS
```

Die Werte sind von 0 für Laufwerk A bis 15 für Laufwerk P durchnummeriert. Das gewählte Laufwerk bleibt nur bis zum nächsten Warmstart erhalten. Soll es auch danach weiterverwendet werden, muß die Nummer für den CCP zusätzlich in den vier niederwertigen Bits der Speicherstelle 0004 hex vermerkt werden:

```
        MOV   A,E
        ANI   15
        MOV   E,A
        LDA   0004H
        ANI   11110000B
        ORA   E
        STA   0004H
```

Dieser Programmcode holt zuerst die Laufwerksnummer ins A-Register und stellt durch ANI 15 sicher, daß der Wertebereich eingehalten wird. Danach überträgt er den Wert wieder ins E-Register und liest das Byte an der Adresse 4. In den höherwertigen Bits ist die User-Nummer codiert. Diese soll natürlich nicht verändert werden. Sie müssen sicherstellen, daß die Laufwerksangabe gelöscht wird. Der zweite ANI-Befehl erledigt dies. Durch ORA E trägt der Prozessor schließlich die neue Laufwerksnummer ein – und mit STA 4 wird sie dauerhaft gespeichert.

Die Funktion 25 stellt fest, welches Laufwerk voreingestellt ist und erledigt damit die entgegengesetzte Arbeit der Funktion 14. Nach dem Aufruf steht im Akkumulator der Laufwerkscode:

```
        MVI   C,25
        CALL  BDOS
        CPI   ...
```

Auch hier können sich Unterschiede zwischen dem Funktionsresultat und dem Inhalt in der Adresse 0004 hex ergeben. Die Laufwerksbezeichnung in 0004 hex läßt sich so ermitteln:

```
        LDA   0004H
        ANI   00001111B
        CPI   ...
```

Die Funktion 28 setzt einen Schreibschutz auf das ausgewählte Laufwerk, so daß ein Schreibversuch einen BDOS-Error mit nachfolgendem Warmstart zur Folge hat. Der Schreibschutz bleibt wirksam, bis ein Warmstart ausgelöst oder das Diskettensystem beziehungsweise die einzelne Diskettenstation neu initialisiert wird. Der Aufruf besteht aus nur zwei Maschinenbefehlen:

```
        MVI   C,28
        CALL  BDOS
```

Der umgekehrte Weg kann auch gewählt werden. Die BDOS-Funktion 29 stellt fest, welche Diskettenlaufwerke schreibgeschützt sind. Im HL-Register steht nach dem Aufruf eine Binärzahl, in der alle geschützten Lauf-

werke auf eins gesetzt sind. Das kennen wir ja schon von der Funktion 24 (Aktive Laufwerke feststellen):

```
MVI   C,29
CALL  BDOS
MOV   A,L
CPI   ...
```

Ihnen ist sicher die Möglichkeit bekannt, mit STAT Dateien dauerhaft vor Schreibzugriffen zu schützen ($R/O statt $R/W) oder aus dem Directory verschwinden zu lassen ($SYS statt $DIR-Typ). Auch ohne das STAT-Programm sind solche Veränderungen mit Hilfe einer BDOS-Funktion durchzuführen. Die Funktion 30 erwartet im DE-Register einen Zeiger auf den File-Control-Block, der die zu verändernde Datei beschreibt. Die Dateimerkmale werden in den Bytes 1 und 2 der File-Extension gespeichert. Ist dort das höchstwertige Bit auf 1 gesetzt, so wird der Schutz wirksam:

- Bit 7 im ersten Byte der Extension bewirkt Schreibschutz
- Bit 7 im zweiten Byte der Extension bewirkt Directory-Schutz

Durch Rücksetzen der Bits im FCB können Sie die Änderungen wieder aufheben. Als Beispiel wollen wir ein Programm schreiben, das Dateien aus dem Directory verschwinden läßt. Der Aufruf soll so aussehen:

```
A>PROTECT d:filename.ext
```

Wir können dabei wieder einmal auf den Standard-FCB an der Adresse 005C hex zurückgreifen:

```
FCB        EQU  005CH
           ORG  TPA
           LXI  SP,STACK
```

Das zweite Byte der Extension befindet sich im FCB an zehnter Stelle:

```
LDA  FCB+10
```

Durch einen ORI-Befehl mit der Zahl 0080 hex setzt der Prozessor das Bit 7 des Werts im Akku auf eins:

```
ORI   0080H
```

Mit einem »Store-Accu-Kommando« speichert der Computer das Byte wieder im FCB ab:

```
STA  FCB+10
```

Der Aufruf der BDOS-Funktion erfolgt mit:

```
MVI  C,30
LXI  D,FCB
CALL BDOS
JMP  WARM
```

Übrigens meldet die Funktion im Akkumulator den Wert 00FF hex, wenn ein Fehler auftritt. Der Einfachheit halber lassen wir diesen Sonderfall aber unberücksichtigt. Lediglich die Stack-Definition müssen wir noch hinzufügen:

```
           DS  20
STACK      EQU $
           END
```

Seien Sie aber vorsichtig bei der Anwendung dieser Funktion, denn manche Programme lassen sich auch mit STAT.COM nicht mehr in einen sichtbaren Zustand bringen. Dies trifft vor allem auf nicht-eindeutige Namen zu – also Namensnennungen, die einen »*« oder ein »?« enthalten.

Die Funktion 35 dient dazu, die Größe einer Datei zu ermitteln:

```
MVI   C,35
LXI   D,FCB
CALL  BDOS
```

Das Ergebnis dieser Berechnung wird in den Bytes 33 und 34 des FCB abgelegt. Die Funktion findet vor allem dann Verwendung, wenn das Programm an das Ende einer Datei gehen soll, um dort zum Beispiel zusätzliche Datensätze anzuhängen. Der FCB zeigt nach dem Funktionsaufruf auf den ersten nicht belegten Satz der Datei. Durch Direktzugriff auf den davorliegenden Record zwingen Sie das BDOS, auch den Zeiger im Byte 32, der bei der sequentiellen Verarbeitung benötigt wird, anzupassen. So können Sie nicht nur direkt auf die Datei zugreifen, sondern auch sequentiell am Dateiende Sätze hinzufügen.

Internes aus CP/M

Die letzte Gruppe der BDOS-Funktionen dient dazu, interne Daten von CP/M dem Benutzer zugänglich zu machen:

Die Funktion 12 meldet im HL-Register die verwendete Betriebssystem-Version. Im H-Register steht, ob es sich um CP/M (Wert 0) oder MP/M (Wert 1) handelt. Im L-Register findet man die Versionsnummer:

Version 1.0 = 00 hex
Version 1.4 = 00 hex
Version 2.0 = 20 hex
Version 2.2 = 22 hex
Version 3.0 = 30 hex
Version 3.1 = 31 hex

Bei den Schneider-Computern liefert CP/M 2.2 auf dem CPC 464, CPC 664 und CPC 6128 im HL-Register den Wert 0022 hex. CP/M Plus auf dem CPC 6128 hingegen 0031 hex. Wenn man genau ist, verwendet der 6128 also nicht CP/M 3.0, sondern CP/M 3.1, auch wenn Cliff Lawson von Amsoft das in einer englischen Fachzeitschrift bestritten hat (Popular Computing Weekly vom 29.8.1985, Seite 6). So lassen sich die verschiedenen Schneider-Computer auch auf der CP/M-Ebene eindeutig unterscheiden. Ein anderer Anwendungsbereich: CP/M kann erst ab der Version 2.0 Direktzugriff-Dateien verwalten. Startversuche auf älteren Versionen lassen sich so im Bedarfsfall abbrechen. Mit dem USER-Befehl wählen Sie im CCP-Prozessor die Benutzerbereiche 0 bis 15. Intern verkraftet CP/M aber die Bereiche 0 bis 31. Mit der Funktion 32 kann man den Benutzerbereich festlegen:

```
MVI   C,32
MVI   E,User
CALL  BDOS
```

Ihre Wahl bleibt bis zum nächsten Warmstart aktiv. Soll sie auch darüber hinaus verwendet werden, müssen Sie wieder die Speicherstelle 0004 hex heranziehen. Die User-Nummer ist in den vier höherwertigen Bits codiert.

Eine zweite Aufgabe der Funktion 32 besteht darin, die aktuelle User-Nummer dem Anwenderprogramm mitzuteilen. Dazu wäre das E-Register mit der Zahl 00FF hex statt der Benutzernummer zu laden. Im Akkumulator findet sich dann das Resultat:

```
MVI   C,32
MVI   E,0FFH
CALL  BDOS
CPI   ...
```

Die beiden letzten BDOS-Funktionen sind nur für sehr spezielle Anwendungen gedacht:

Funktion 27 liefert im HL-Register einen Zeiger auf die Belegungstabelle der aktiven Diskette. Dieser RAM-Block wird vom Betriebssystem immer auf dem neuesten Stand gehalten:

```
MVI   C,27
CALL  BDOS
```

Funktion 31 meldet im HL-Register die Adresse des DPB-Blocks der Diskette. »DPB« heißt »Disk Parameter Block«. In diesem Speicher sind alle wichtigen Informationen über den Diskettenaufbau festgehalten, wie Zahl der Sektoren und Spuren, Länge der Sektoren und die Maximalzahl der Directory-Einträge.

```
MVI   C,31
CALL  BDOS
```

Direkt hinein ins BIOS

Damit kennen Sie jetzt alle BDOS-Funktionen. In Bild 4 sind sie noch einmal aufgeführt. Bild 5 zeigt alle CP/M-CTRL-Funktionen. Wenn Sie sich fit genug fühlen, noch mehr Informationen aufzunehmen, können Sie weiterlesen und erfahren, wie sich BIOS-Funktionen aufrufen lassen und was sie leisten. Andernfalls machen Sie jetzt erst einmal eine Pause.

Bisher sind wir davon ausgegangen, daß ein Systemteil von CP/M immer nur auf den unmittelbar darunter liegenden Teil zugreifen kann, zum Beispiel das BDOS auf das BIOS, das Anwenderprogramm auf das BDOS, der CCP auf das BDOS und so weiter. Doch es gibt auch »Abkürzungen«. Sie können in Ihren Programmen auch direkt die BIOS-Funktionen aufrufen.

BDOS-Funktionen werden über die Speicherstelle 0005 hex aufgerufen. Anhand ihrer Funktionsnummer werden sie decodiert. BIOS-Funktionen hingegen werden über eine Sprungtabelle aktiviert, in der für jede BIOS-Funktion ein drei Byte langer Sprungbefehl reserviert ist. Diese Sprungtabelle (siehe unten) beginnt beim nicht ausgebauten CPC unter CP/M 2.2 an der Adresse AD00 hex.

Rufen Sie aber niemals die BIOS-Routinen anhand dieser Adressen auf.

AD00 hex	– Funktion 0:	BOOT (Kaltstart)
AD03 hex	– Funktion 1:	WBOOT (Warmstart)
AD06 hex	– Funktion 2:	CONST (Tastaturstatus)
AD09 hex	– Funktion 3:	CONIN (Tastatureingabe)
AD0C hex	– Funktion 4:	CONOUT (Bildschirmausgabe)
AD0F hex	– Funktion 5:	LIST (Druckerausgabe)
AD12 hex	– Funktion 6:	PUNCH (Lochstreifen stanzen)
AD15 hex	– Funktion 7:	READER (Lochstreifen lesen)
AD18 hex	– Funktion 8:	HOME (Diskettenkopf auf Track 0)
AD1B hex	– Funktion 9:	SELDSK (Diskettenlaufwerk wählen)
AD1E hex	– Funktion 10:	SETTRK (Track selektieren)
AD21 hex	– Funktion 11:	SETSEC (Sektor selektieren)
AD24 hex	– Funktion 12:	SETDMA (DMA-Adresse setzen)
AD27 hex	– Funktion 13:	READ (Record lesen)
AD2A hex	– Funktion 14:	WRITE (Record schreiben)
AD2D hex	– Funktion 15:	LISTST (Druckerstatus)
AD30 hex	– Funktion 16:	SECTRAN (Recordnummer übersetzen)

Sprungtabelle beim einfachen CPC unter CP/M 2.2

Sie sind von Computer zu Computer verschieden – und falls Sie irgendwann einmal Ihren RAM-Speicher ausbauen, können Sie alle Programme, die auf einen festen Beginn der BIOS-Sprungtabelle zugreifen, wegwerfen.

Es gibt aber einen Weg, die BIOS-Routinen geräteunabhängig zu nutzen. In der Adresse 0000 hex, die wir bisher schon für den Warmstart benutzt haben, steht ein Sprung-Befehl auf den BIOS-Block – genauer auf die BIOS-Funktion 1 (Warmstart). Die exakte Startadresse der Sprungtabelle liegt also drei Byte darunter.

Die Mikroprozessoren der 80er-Reihe legen Sprungbefehle immer in diesem Format ab:

1. Byte: C3H (JMP)
2. Byte: Lowbyte der Adresse
3. Byte: Highbyte der Adresse

Mit LHLD 1 holen Sie sich also die Startadresse der BIOS-Warmstart-Routine ins HL-Register. Drei DCX-Befehle sorgen dafür, daß Sie den Tabellenanfang erreichen:

```
LHLD 1
DCX  H
DCX  H
DCX  H
```

Um eine BIOS-Routine anhand ihrer Funktionsnummer aufzurufen, ist folgender Programmcode empfehlenswert:

```
         LHLD 1
         DCX  H
         DCX  H
         DCX  H
         LXI  D,Nummer*3
         DAD  D
         LXI  D,RETADDR
         PUSH D
         PCHL

RETADDR EQU  $
```

Die Vorschriften bei der Übergabe von Registerwerten sind genauso streng gefaßt wie bei den BDOS-Routinen:

- Die Übergabe von 8-Bit-Werten erfolgt im C-Register, die von 16-Bit-Daten im BC-Doppelregister.
- Das BIOS meldet 8-Bit-Werte im Akkumulator, 16-Bit-Werte im HL-Doppelregister.

Gehen wir die Funktionen des BIOS der Reihe nach durch:

Benötigte Register: keine
Gemeldete Register: kehrt nicht zum Programm zurück.

Diese Funktion setzt den Computer zurück und lädt das Betriebssystem neu, so als hätte der Benutzer einen Reset ausgelöst und unter Basic den RSX-Befehl »ICPM« eingegeben. Nützlich ist die Funktion, um vom CCP aus neue Disketten einzuloggen und gleichzeitig den System- und Boot-Sektor nachzuladen:

```
ORG   0100H
LHLD  1
DCX   H
DCX   H
DCX   H
PCHL
END
```

Wenn Sie das Programm zum Beispiel als »COLD.COM« speichern, können Sie jederzeit Disketten wechseln und gleichzeitig die Boot-Sektoren neu laden. So ist es etwa möglich, zwischen der Standard-CP/M-Tastaturbelegung und der von Wordstar hin- und herzuwechseln, ohne dauernd CTRL-SHIFT-ESC zu drücken.

WBOOT – Warmstart:
Benötigte Register: keine
Gemeldete Register: kehrt nicht zum Programm zurück

Diese Funktion lädt das BDOS und den CCP neu von der Diskette. JMP 0 oder die BDOS-Funktion 0 sind aber einfacher zu handhaben.

CONST – Tastaturstatus:
Benötigte Register: keine
Gemeldetes Register: Akkumulator

Die Funktion 2 fragt die Tastatur ab. Wenn keine Taste gedrückt wurde, ist der Akkuinhalt Null, andernfalls 00FF hex.

CONIN – Tastatureingabe:
Benötigte Register: keine
Gemeldetes Register: Akkumulator

Die BIOS-Funktion 3 wartet auf einen Tastendruck und liefert den ASCII-Code der Taste im Akkumulator.

CONOUT – Bildschirmausgabe:
Benötigtes Register: C-Register
Gemeldete Register: keine

Wenn sich im C-Register der Wert eines ASCII-Zeichens befindet, wird das Zeichen auf dem Bildschirm ausgegeben.

LIST – Druckerausgabe:
Benötigtes Register: C-Register
Gemeldete Register: keine

Das Zeichen im C-Register wird an den Druckerausgang LST: geschickt.

PUNCH – Lochstreifen stanzen:
Benötigtes Register: C-Register
Gemeldete Register: keine

Das Zeichen im C-Register wird vom BIOS an den Lochstreifen-Stanzer abgeschickt. Die BDOS-Funktion 4 ist aber aus Gründen der Kompatibilität mit anderen CP/M-Versionen vorzuziehen.

READER – Lochstreifen lesen:
Benötigte Register: keine
Gemeldetes Register: Akkumulator

Die Funktion wartet auf eine Eingabe vom Lochstreifenleser und übermittelt das Zeichen dem aufrufenden Programm im Akkumulator. Auch hier ist die Verwendung der entsprechenden BDOS-Funktion anzuraten.

HOME – Diskettenkopf auf Track 0:
Benötigte Register: keine
Gemeldete Register: keine

Der Schreib- und Lesekopf der aktiven Diskettenstation wird auf den Track (Spur) 0 bewegt. Die Funktion ist für Anwender kaum von Interesse, da das BDOS beim Diskettenzugriff die Positionierung des Schreib-/Lesekopfes automatisch durchführt.

SELDSK – Diskettenlaufwerk wählen:

Benötigtes Register: C-Register
Gemeldetes Register: HL-Register

Das Diskettenlaufwerk, dessen Nummer sich im C-Register befindet, wird ausgewählt. Die Werte sind 0 für das Laufwerk A, 1 für das Laufwerk B, bis 15 für das Laufwerk P. Im HL-Register steht nach dem Aufruf dieser BIOS-Funktion ein Zeiger auf den DPH (Disk Parameter Header). Dies ist eine Tabelle, die das Betriebssystem zur internen Verwaltung benötigt. Wenn das angsprochene Laufwerk nicht verfügbar ist, enthält HL den Wert 0.

SETTRK – Track selektieren:
Benötigtes Register: BC-Register
Gemeldete Register: keine

Die Spur, deren Nummer im BC-Register steht, wird angewählt. Beim Schneider-CPC sind Werte zwischen 0 und 39 gestattet.

SETSEC – Sektor selektieren:
Benötigtes Register: BC-Register
Gemeldete Register: keine

Der Sektor, dessen Name das BC-Register enthält, wird selektiert. Damit führt das BIOS aber den eigentlichen Diskettenzugriff noch nicht durch.

SETDMA – DMA-Adresse setzen:
Benötigtes Register: BC-Register
Gemeldete Register: keine

Die Adresse des 128-Byte-Speicherbereichs, der bei Diskettenoperationen als Zwischenspeicher dient, wird auf den Wert im BC-Register festgelegt. Verwenden Sie aber besser die BDOS-Funktion 26.

```
-D0080,00FF
0080 09 20 46 49 4C 45 2E 43 4F 4D 00 4D 00 00 00 02 . FILE.COM.M....
0090 5C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 \...............
00A0 E5 52 45 41 44 43 4D 44 20 50 52 4E 00 00 00 04 .READCMD PRN....
00B0 5E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ^...............
00C0 E5 52 45 41 44 43 4D 44 20 48 45 58 00 00 00 01 .READCMD HEX....
00D0 5F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 _...............
00E0 E5 52 45 41 44 43 4D 44 20 43 4F 4D 00 00 00 01 .READCMD COM....
00F0 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 `...............
```

Bild 1. Die Zeropage im Bereich 0080 bis 00FF hex

```
-D005C,007F
005C 00 46 49 4C                                     .FIL
0060 45 20 20 20 20 43 4f 4d 00 00 00 26 00 20 20 20 E    COM...&.
0070 20 20 20 20 20 20 20 20 00 00 00 00 00 00 00 00      ......
```

Bild 2. Die Zeropage im Bereich 005C bis 007F hex

READ – Record lesen:
Benötigte Register: keine
Gemeldetes Register: Akkumulator

Der Sektor, der durch die BIOS-Funktionen SELDSK, SETTRK und SETSEC spezifiziert wurde, wird in den Speicher gelesen. Die Startadresse muß dazu durch die BDOS-Funktion 26 oder die BIOS-Funktion 12 gesetzt werden. Eventuelle Fehler zeigt der Akkumulator an. Ist der Akku gleich Null, trat kein Fehler auf, bei A=1 ist jedoch etwas schiefgegangen.

WRITE – Record schreiben:
Benötigte Register: keine
Gemeldetes Register: Akkumulator

Die BIOS-Funktion 14 schreibt den Sektor, der durch SELDSK, SETTRK und SETSEC angegeben wurde, auf die Diskette. Die DMA-Adresse muß vorher wie bei der BIOS-Funktion 13 festgelegt werden. Fehlerbedingungen werden im Akku angezeigt – ebenfalls entsprechend der Funktion 13.

LISTST – Druckerstatus:
Benötigte Register: keine
Gemeldetes Register: Akkumulator

Diese Funktion fragt ab, ob der Drucker empfangsbereit ist und ein Zeichen übernehmen kann. Abhängig

```
************************************************
*                                              *
* ERASE.COM -- Löschen mit Sicherheitsabfrage *
*                                              *
************************************************

BDOS      EQU  0005H
TPA       EQU  0100H
WARM      EQU  0000H
FCB       EQU  005CH

          ORG  TPA

START     LXI  SP,STACK

LAUFWERK  LDA  FCB
          ADI  64
          MOV  E,A
          MVI  C,2
          CALL BDOS

DATEINAME LXI  H,FCB+1
          MVI  A,11

SCHLEIFE  MOV  E,M
          PUSH PSW
          PUSH H
          MVI  C,2
          CALL BDOS
          POP  H
          POP  PSW
          INX  H
          DCR  A
          JNZ  SCHLEIFE

ABFRAGE   LXI  D,SICHER$
          MVI  C,9
          CALL BDOS
          MVI  C,1
          CALL BDOS
          CPI  'J'
          JNZ  WARM

ERASE     MVI  C,19
          LXI  D,FCB
          CALL BDOS
          JMP  WARM

SICHER$   DB   '- Sind Sie sicher? $'
          DS   20
STACK     EQU  $

          END
```

Bild 3. Ein nützliches CP/M-Programm: ERASE.COM

davon ist der Akkumulator gleich Null (Drucker nicht empfangsbereit) oder 00FF hex (Drucker empfangsbereit).

SECTRAN – Recordnummer übersetzen:

Benötigtes Register: BC-Register
Gemeldetes Register: HL-Register

Die Routine übersetzt die vom BDOS verwendete logische Sektornummer in physikalische, Hardware-abhängige. Ins BC-Register ist die Sektornummer zu laden, zurückgegeben wird die physikalische Sektornummer im HL-Register.

Damit endet unser Kurs, der Sie in die Programmierung unter CP/M 2.2 auf dem Schneider einführen sollte. Falls Sie mehr wissen wollen: Es gibt eine ganze Menge Spezialliteratur.

(Martin Kotulla/hg)

CTRL A	– Cursor um ein Zeichen nach links setzen (nur CP/M plus)
CTRL B	– Cursor an den Anfang der Zeile setzen (oder an das Ende, wenn er am Anfang steht) (nur CP/M plus)
CTRL C	– Warmstart ausführen
CTRL E	– Cursor in die nächste Zeile setzen
CTRL F	– Cursor um ein Zeichen nach rechts setzen (nur CP/M plus)
CTRL G	– Zeichen unter dem Cursor löschen (nur CP/M plus)
CTRL H	– Zeichen links vom Cursor löschen
CTRL I	– Cursor zur nächsten TAB-Position setzen
CTRL J	– Befehl ausführen (ENTER)
CTRL K	– Zeile zwischen Cursor und Zeilenende löschen (nur CP/M plus)
CTRL M	– Befehl ausführen (ENTER)
CTRL P	– Drucker ein-/ausschalten
CTRL Q	– Scrollen nach CTRL-S wieder einschalten
CTRL R	– Befehlszeile wiederholen
CTRL S	– Ausgabe auf dem Bildschirm unterbrechen
CTRL U	– Zeile löschen
CTRL W	– Zeile wieder anzeigen (nur CP/M plus)
CTRL X	– Zeile löschen
CTRL Z	– String-Ende bei PIP und ED

Bild 5. Alle CTRL-Codes auf einen Blick

Funktion (Register C)	Beschreibung
0	System-Neustart (system reset) Eingabe: – Ausgabe: – Rücksprung ins CCP und CP/M-Warmstart.
1	Konsolen-Eingabe (console input) Eingabe: – Ausgabe: A (Zeicheneingabe) Lesen des nächsten Zeichens von der Tastatur und Prüfen auf CP/M-Kontroll-Zeichen.
2	Konsolen-Ausgabe Eingabe: E (Zeichen auf den Bildschirm) Ausgabe: – Schreiben eines Zeichens auf den Bildschirm.
3	Lochstreifen lesen (reader input) Eingabe: – Ausgabe: A (Zeichen lesen) Bei der Kopplung zweier Rechner kann mit dieser Funktion ein Zeichen gelesen werden.
4	Lochstreifen stanzen (punch output) Eingabe: E (Zeichen stanzen) Ausgabe: – Bei der Kopplung zweier Rechner kann mit dieser Funktion ein Zeichen ausgegeben werden.
5	Drucker-Ausgabe (list-output) Eingabe: E (Zeichen drucken) Ausgabe: – Schreiben eines Zeichens auf dem Drucker.
6	Direkte Konsolen Ein-/Ausgabe (direct console I/O) Eingabe: E (Zeichen auf den Bildschirm/Ausgabe) E FF hex (Eingabe) Ausgabe: A Status (Ausgabe) A Zeichen (Eingabe) Ausführen einer Konsolen-Eingabe (von der Tastatur) und Konsolenausgabe (auf den Bildschirm) und Übertragen ohne Prüfen oder Korrigieren durch BDOS.
7	I/O-Byte holen (get (I/O byte) Eingabe: – Ausgabe: A (I/O-Byte)
8	I/O-Byte setzen (set I/O byte) Eingabe: E (neues I/O-Byte) Ausgabe: –
9	Ausgabe eines Strings (print string) Eingabe: DE (String-Adresse) Ausgabe: – Schreiben einer Zeichenkette (String) auf dem Bildschirm (Konsole). Das Ende des Strings wird durch das Zeichen »$« angezeigt.
10	Eingabe des Konsolen-Puffers (read console buffer) Eingabe: DE (Pufferstartadresse) Ausgabe: Zeichen im Puffer Eingabe der über die Tastatur in einen Puffer eingegebenen Zeichen (Textzeile). Die Eingabe einer Zeile wird entweder bei Erreichen der Maximalzeichenzahl abgebrochen, oder wenn das Zeichen »CR« oder »LF« eingegeben wurde.
11	Konsolen-Status holen (get console status) Eingabe: – Ausgabe: A (Konsolen-Status) Prüfen des Tastatur-Status auf Eingabe eines Zeichens. A enthält 01 hex (nicht FF), wenn ein Zeichen bereitsteht, oder 00 hex, wenn nicht.
12	CP/M-Version ermitteln (return version number) Eingabe: – Ausgabe: HL (Versionsnummer)
13	Rücksetzen des Diskettensystems (reset disk system) Eingabe: – Ausgabe: – Zurücksetzen des gesamten Diskettensystems in den Anfangszustand.
14	Bezugslaufwerk selektieren (select disk) Eingabe: E (ausgewählte Laufwerksnummer) Ausgabe: – Selektieren des Bezugsdiskettenlaufwerkes (A = 0 und B = 1)
15	Eröffnen einer Datei (open file) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Eröffnen einer Disketten-Arbeitsdatei. Gibt 255 (FF hex) in A zurück, wenn die Datei nicht gefunden werden konnte.
16	Schließen einer Datei (close file) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Schließen einer Diskettendatei. Gibt 255 (FF hex) in A zurück, wenn eine Datei nicht gefunden werden konnte.
17	Suche nach erstem Namen (search for first) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Suchen nach der ersten Datei, deren Name auf die Angabe im FCB paßt. Gibt 255 (FF hex) in A zurück, wenn kein Eintrag gefunden wurde.
18	Suchen nach nächstem Eintrag (search for next) Eingabe: –

Bild 4. Alle BDOS-Funktionen auf einen Blick.

Funktion (Register C)	Beschreibung
	Ausgabe: A (Directory-Code) Durch die Funktion 17 wird das erste Auftreten gesucht, hier wird nun der nächste Eintrag geliefert. Gibt ebenfalls 255 (FF hex) in A zurück, wenn kein Eintrag gefunden wurde.
19	Löschen einer Datei (delete file) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Gibt 255 (FF hex) in A zurück, wenn keine Datei gefunden werden konnte.
20	Sequentielles Lesen (read sequential) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Lesen des nächsten 128-Byte-Record (Aufzeichnung) in den Arbeitsspeicher, beginnend von der aktuellen DMA-Adresse. Gibt bei erfolgreicher Operation 00 hex in A zurück; jedoch ungleich Null, wenn das Dateiende gefunden wurde.
21	Sequentielles Schreiben (write sequential) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Schreiben des nächsten 128-Byte-Record (Aufzeichnung) in die durch FCB spezifizierte Datei; Beginn an der aktuellen DMA-Adresse. Gibt 00 hex in A bei erfolgreicher Operation zurück, jedoch ungleich Null bei einer vollen Diskette.
22	Datei erzeugen (make file) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Erzeugen einer neuen, aber leeren Diskettendatei mit dem im FCB angegebenen Namen. Meldet 255 (FF hex) in A, wenn die Operation nicht möglich war, zum Beispiel bei voller Directory.
23	Datei umbenennen (rename file) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Umbenennen des Namens einer Diskettendatei. Der alte Name befindet sich in den ersten 16 Byte des FCB, der neue Name in den nächsten 16 Byte. Meldet 255 (FF hex) in A, wenn die Umbenennung nicht möglich war.
24	Verfügbare Laufwerke ermitteln (return login vector). Eingabe: – Ausgabe: HL (Login-Vektor) Meldet den Login-Vektor (Laufwerksvektor). Bit 0 von Register L entspricht dem Laufwerk A, Bit 1 dem Laufwerk B und so weiter. Für jedes aktive Laufwerk steht eine 1 und für jedes inaktive eine Null (0).
25	Aktuelles Laufwerk melden (return current disk) Eingabe: – Ausgabe: A (Nummer des aktuellen Laufwerks) Meldet die Nummer des aktuellen (Bezugs-) Disketten-Laufwerks (0 = A und 1 = B).
26	DMA-Adresse festlegen (get DMA address) Eingabe: DE (DMA-Adresse) Ausgabe: – Vermerkt die Pufferadresse (DMA – direct memory address) des ab dieser Adresse beginnenden 128-Byte-Disketten-Sektor-Puffers.
27	Belegungsverzeichnis holen (get addr-alloc) Eingabe: – Ausgabe: HL (ALLOC-Adresse) Meldet die Adresse einer Tabelle, die die belegten Blöcke der Diskette im aktuellen (Bezugs-) Laufwerk darstellt.
28	Diskette auf Schreibschutz setzen (write protect disk) Eingabe: – Ausgabe: – Setzt den Schreibschutz für die Bezugsdiskette.
29	Schreibgeschützte Laufwerke (get read only vector) Eingabe: – Ausgabe: HL (Schreibschutzvektor) Meldet den Schreibschutzvektor, der anzeigt, welches Laufwerk gerade schreibgeschützt ist. Bit 0 des Registers L ist Laufwerk A, Bit 1 Laufwerk B und so weiter. Eine 1 bedeutet, daß das Laufwerk schreibgeschützt ist.
30	Datei-Attribute setzen (set file attribute) Eingabe: DE (FCB-Adresse) Ausgabe: A (Directory-Code) Setzt die Datei-Attribute auf Schreibschutz und System-Datei.
31	Adresse der Laufwerksparameter holen (get addr-disk parms) Eingabe: – Ausgabe: HL (DPB-Adresse) Meldet die Ausgabe des Disketten-Parameter-Blocks DPB.
32	USER-Code setzen/holen (set/get user code) Eingabe: E (User-Code setzen) E FF hex (User-Code holen) Ausgabe: A (User-Code holen) Holt oder setzt den aktivierten User-Code (Benutzerbereich).
33	Lesen mit wahlfreiem Zugriff (read random) Eingabe: DE (FCB-Adresse) Ausgabe: A (Fehlercode) Lesen einer Disketten-Datei-Aufzeichnung (record). In A steht ein Fehlercode: 0 = kein Fehler 1 = Leseversuch unbeschriebener Daten 3 = kein Abschluß im aktivierten Bereich möglich 4 = Versuch, einen unbeschriebenen Bereich anzuwählen 6 = Versuch, über das Diskettenende zu positionieren.
34	Schreiben mit wahlfreiem Zugriff (write random) Eingabe: DE (FCB-Adresse) Ausgabe: A (Fehlercode) Schreiben einer Disketten-Datei-Aufzeichnung (record). Der Fehlercode ist identisch zu Funktion 33, mit folgender Erweiterung: 5 = außerhalb des Directory-Bereiches.
35	Berechnen der Datei-Größen (compute file size) Eingabe: DE (FCB-Adresse) Ausgabe: Dateigröße Legt die Dateigröße, als Anzahl der Records, in den letzten drei Byte (33, 34, 35) des FCB, im »random record file« ab.
36	Random-Record setzen (set random record) Eingabe: DE (FCB-Adresse) Ausgabe: – Setzt die Random-Record-Nummer einer sequentiellen Aufzeichnung (record). Die Random-Record-Nummer ist in den letzten drei Byte (33, 34, 35) des FCB, im »random record field«, abgelegt.
37	Laufwerk(e) zurücksetzen (reset drive) Eingabe: DE (Laufwerksvektor) Ausgabe: – Setzt die im Laufwerksvektor angegebenen Laufwerke zurück. Bit 0 von Register E steht für Laufwerk A, Bit 1 für Laufwerk B und so weiter. Bei »1« wird das entsprechende Laufwerk zurückgesetzt.
38	nicht verwendet
39	nicht verwendet
40	Ein mit Null gefülltes Random-Record schreiben (write random with zero fill) Eingabe: DE (FCB-Adresse) Ausgabe: A (Fehlercode) Identisch mit Funktion 34 (Schreiben mit wahlfreiem Zugriff), außer daß die neuen Blocks mit Null (0) gefüllt werden.

Z80 kontra 8080

Ursprünglich war CP/M für den Intel-Prozessor 8080 gedacht. Doch auch der Z80 kommt mit dem 8080-Standard zurecht. Da gibt es Gemeinsamkeiten – und viele Unterschiede.

Wollte man den Intel-8080 mit dem Z80-Prozessor in einen Wettbewerb schikken, wäre das mehr als unfair. Denn der Z80-Chip erschien erst Jahre nach der 8080-CPU. In ihn konnten deshalb alle positiven Eigenschaften des »Erstgeborenen« integriert und die Schwächen soweit wie möglich ausgebügelt werden. Dennoch ist es für Z80-Programmierer, die auf CP/M umsteigen wollen, wichtig, auch die Eigenschaften des Konkurrenten zu kennen.

Neue Mnemonics

Der augenfälligste Unterschied liegt in den mnemonischen (Mnemonik (gr.): Kunst, das Einprägen von Gedächtnisstoff durch besondere Lernhilfen zu erleichtern) Befehlen beider Prozessoren. Allerdings sind beide objektcodekompatibel. Das heißt, daß bei beiden Chips beispielsweise der Befehl, der den Akkumulator mit einer Zahl lädt, den Code 3E hex besitzt. Dabei heißt der 8080-Befehl »MVI A,n«, der des Konkurrenten aber »LD A,n«. Wer diese Tatsache verstehen will, muß sich ein paar Jahre in die Vergangenheit zurückversetzen; genauer in die Jahre vor 1974, in die »Computer-Steinzeit« sozusagen. Damals arbeitete ein Mr. Faggin bei Intel. Eine seiner Großtaten war die Entwicklung des 8080-Mikroprozessors. 1974 trennte sich Faggin von der Firma und machte sich selbständig. Er nannte seine neue Firma »Zilog«. Sein wichtigstes Projekt wurde die Entwicklung eines 8080-kompatiblen Mikrochips. Nach einiger Zeit brachte Zilog tatsächlich einen Prozessor heraus, den Faggin auf den Namen »Z80« taufte.

Die Begründung dafür, daß beide Prozessoren trotz Kompatibilität unterschiedliche Befehlsnamen verwenden, findet man in den amerikanischen Urheberrechtsgesetzen. Hatte Faggin so viel Dreistigkeit, die gleichen Maschinencodes wie die des Intel-8080 zu verwenden, so verließ ihn der Mut, als es daran ging, diesen Befehlen Namen zu geben. Sicher erkannte er aber auch, daß die Befehlsnamen des Intel-Produkts viel zu umständlich gewählt waren und wollte nun einen möglichst linearen Befehlssatz verwenden.

Das wohl am häufigsten herangezogene, da deutlichste, Vergleichsobjekt ist der Ladebefehl. Dieser heißt beim Z80 für alle möglichen Ladevorgänge einheitlich »LD« für »Load«. Der 8080-Prozessor benutzt hingegen eine Vielzahl von unterschiedlichen Befehlsnamen, zum Beispiel MOV, MVI, LXI, LHLD, SHLD, STA, STAX, LDA, LDAX und so weiter. Für jeden Einzelfall gibt es beim 8080-Prozessor einen speziellen Namen. Tabelle 1 zeigt die Unterschiede bei diesem Befehl. Auch sind die 8080-Konstrukteure nicht allzu konsequent an die Bezeichnung der Doppelregister herangegangen. Genauso wie beim Z80 heißen sie zwar BC, DE und HL, beim Mnemo-Code werden sie allerdings manchmal zu B, D und H abgekürzt. Es gibt also Befehle wie LXI, DCX und INX, die den »amputierten« Registernamen verwenden, (beispielsweise INX H) und andere, die den kompletten, aus zwei Buchstaben bestehenden, Namen gebrauchen (LHLD, SHLD, PCHL und XTHL).

Intel-8080		Zilog Z80
LDA	adr	LD A, (adr)
LDAX	B	LD A, (BC)
LDAX	D	LD A, (DE)
LHLD	adr	LD HL, (adr)
LXI	B,word	LD BC,word
LXI	D,word	LD DE,word
LXI	H,word	LD HL,word
MOV	regz,regq	LD regz,regq
MOV	A,M	LD A, (HL)
MOV	M.A	LD (HL), A
MVI	regz,byte	LD regz,byte
MVI	M,byte	LD (HL),byte
SHLD	adr	LD (adr) ,HL
SPHL		LD SP,HL
STA	adr	LD (adr) ,A
STAX	B	LD (BC) ,A
STAX	D	LD (DE) ,A

adr = Adresse
word = 16-Bit-Zahl
byte = 8-Bit-Zahl
regz = Zielregister
regq = Quellregister

Tabelle 1.
Zwei »Philosophien« bei Mnemonics – die Ladebefehle im Vergleich

Behandelt man die 8080-Doppelregister als Adreßzeiger, so findet man recht willkürliche Unterschiede bei den Bezeichnungen. Die Z80-Befehle LD A,(BC), LD A,(DE) und LD A,(HL) mutieren zu LDAX B, LDAX D und MOV A,M. Diese Inkonsequenz rührt von der Geschichte der Intel-Prozessoren her. Denn vor dem 8080 hatte Faggin schon den Intel-8008 entwickelt. Dieser ist weit weniger komfortabel. Soll beispielsweise der Akkumulator mit dem Inhalt einer Speicherstelle geladen werden, so muß man das HL-Register mit einem Zeiger auf die gewünschte Adresse laden. Erst ein zweiter Befehl holt dann den Inhalt der Speicherstelle in den Akku. Da das HL-Register regelmäßig zur Kommunikation mit dem Speicher benutzt wird, spendierte man ihm einen zusätzlichen Namen. Das M-Register (»M« steht für »Memory«) wurde als die Adresse definiert, auf die das HL-Register zeigt. Da der 8080 die ganze Firmengeschichte von Intel mit sich herumschleppen muß, entstanden die teilweise völlig abwegigen Bezeichnungen.

Ein schöner Name macht's noch nicht

Natürlich war auch Faggin klar, daß er den Markt nicht allein mit »schöneren« Bezeichnungen für dieselben Befehle für sich gewinnen konnte. So überlegte er, was ihm bei der Arbeit mit der 8080-CPU besonders gefehlt hatte und verbesserte seinen Z80 entsprechend.

Zuerst einmal stattete er den neuen Chip mit einem zusätzlichen Register aus – dem Indexregister IX. Dieses eignet sich besonders gut zur Verarbeitung von Tabellen. Man lädt IX mit einem Zeiger auf den Tabellenanfang und greift dann über ein »Displacement« (Verschiebewert) auf die einzelnen Bytes zu:

```
LD  IX,Table
LD  A,(IX+2)   2 Byte ab
               Tabellenanfang
ADD A,(IX+55) 55 Bytes vom
               Tabellenanfang
```

Da ihm diese Idee so gut gefiel, begnügte er sich nicht mit IX, sondern stellte dem neuen Register einen Zwillingsbruder nämlich das Register IY, zur Seite. So können Z80-Programmierer sehr einfach mit zwei Tabellen gleichzeitig arbeiten.

Intel 8080:

Akku	Flag
B-Register	C-Register
D-Register	E-Register
H-Register	L-Register
SP-Register	
PC-Register	

Tabelle 2. Die Register des 8080 und Z80 im Vergleich. Deutlich erkennbar ist die Vielfalt der Z80-Register, die komfortableres Programmieren erlauben

Zilog Z80:

Akku	Flag
B-Register	C-Register
D-Register	E-Register
H-Register	L-Register
SP-Register	
PC-Register	
Indexregister IX	
Indexregister IY	
R-Register	I-Register

Hauptregistersatz

A'-Register	F'-Register
B'-Register	C'-Register
D'-Register	E'-Register
H'-Register	L'-Register

Zweiter Registersatz

Aber auch das war ihm noch nicht genug, und so schuf er einen zweiten Registersatz: Zu AF, BC, DE und HL gesellten sich AF', BC', DE' und HL'. Mittels zweier einfacher Befehle (EX AF, AF' und EXX) können die Programmierer zwischen den beiden Registersätzen wählen. Leider ist diese leistungsfähige Erweiterung beim Schneider-CPC nur sehr beschränkt nutzbar. Die Zweitregister werden nämlich von den Interrupt-Routinen, die die Maschinenprogramme regelmäßig unterbrechen, verwendet. Sie ohne Vorkehrungen zu verändern, hat deshalb meist katastrophale Folgen – in aller Regel den Absturz des Systems. Wer nun aber gar nicht auf die zusätzlichen Register verzichten will, findet im Anhang zum Firmware-Handbuch Hilfe (wird von Schneider vertrieben und ist jedem Maschinenprogrammierer dringendst zu empfehlen).

Der einfachste Weg: unterdrücken von Interrupts mit DI – Umschalten auf Zweitregister – sichern der Register auf dem Stack – Abarbeitung des Programms – Wiederherstellung der Register – Umschalten auf den normalen Registersatz – Zulassen von Interrupts mit EI und Rücksprung aus dem Programm. Wen dieser Telegrammstil überfordert, der sollte vorerst auf die Arbeit mit dem Zweitregister verzichten. Denn ganz so einfach ist das ganze nicht, da die Interrupt-Routinen äußerst wichtige Funktionen wie automatische Tastaturabfrage, Abschalten des Tongenerators, Steuerung des Floppymotors und so weiter, übernehmen. Es ist nämlich nicht sehr schön, den Motor des Laufwerks unter Umständen mehrere Minuten laufen zu lassen, ohne daß der Computer wirklich auf die Diskettenstation zugreift.

Wohl vor allem für Entwickler von Systemsoftware ist der IM-Befehl von Interesse, der verschiedene Methoden zur Bearbeitung von Interrupts gestattet. Hierzu gehören auch wieder zwei neue Register: das Interrupt- und das Refresh-Register. Das I-Register liefert das Highbyte der Adresse der Interrupt-Routine im Interrupt-Modus 2, während das R-Register für den automatischen Refresh der RAM-Bausteine sorgt. Einen Vergleich des Registeraufbaus 8080 – Z80 zeigt die Tabelle 2. Zur Rückkehr aus einer Interrupt-Routine dienen RETI (Return from Interrupt) und RETN (Return from NMI-Interrupt).

Neue Arithmetikbefehle für den Z80

Die übrigen Erweiterungen des Z80 betreffen den Befehlssatz und erheben ihn damit zu einem der leistungsfähigsten 8-Bit-Prozessoren überhaupt. Neben der vorzeichenlosen Addition von 16-Bit-Registern (ADD HL,BC) gibt es auch eine vorzeichenbehaftete Version (ADC HL,BC). Ebenso ist die vorzeichenbehaftete Subtraktion von Doppelregistern erlaubt: SBC HL,BC; SBC HL,DE; SBC HL,HL und SBC HL,SP. Ganz allgemein ist die Arbeit mit den Doppelregistern bedeutend einfacher. Nicht nur der 16-Bit-Akku HL kann mit dem Speicher kommunizieren, auch die Registerpaare BC, DE und SP sind jetzt dazu fähig. LD BC, (nnnn); LD (nnnn), BC; LD DE, (nnnn); LD (nnnn), DE; LD SP, (nnnn) und LD (nnnn), SP nehmen damit dem Programmierer sehr viel Routinearbeit ab.

Im Telegrammstil die restlichen Erweiterungen: Befehle zum gezielten Testen, Löschen und Setzen von Bits in allen Registern (BIT, RES und SET) und relative Sprünge, sowohl unbedingt als auch bedingt an das Flag-Register geknüpft. Diese bedingten Relativsprünge dürfen sich allerdings nur auf das Carry-Flag und das Zero-Flag beziehen. Außerdem erleichtert DJNZ in Verbindung mit dem B-Register die Konstruktion von Programmschleifen.

Leistungsstark durch Blockbefehle

Zusätzlich integrierte Faggin eine Vielzahl von Blockbefehlen (Blocksuch-, Blockkopier- und Blockein- und -ausgabebefehle): CPD, CPDR, CPI (nicht zu verwechseln mit dem 8080-Vergleichsbefehl), CPIR, LDD, LDDR, LDI, LDIR, IND, INDR, INI, INIR, OUTD, OTDR, OUTI oder OTIR. Diese hardwaremäßig implementierten Befehle sind natürlich jeder Softwarelösung an Geschwindigkeit haushoch überlegen.

Bezüglich der Ports bleibt noch zu sagen, daß die IN- und OUT-Kommandos jetzt mit allen Registern arbeiten. Leider schränkt die Hardware der Schneider-Computer die Verwendung dieser Portbefehle stark ein. Da zur Decodierung der Portnummer nicht 8 Bit, sondern 16 Bit verwendet werden, muß neben dem C-Register auch das B-Register zur Adreßbildung herhalten und kann deswegen nicht mehr als Schleifenzähler der Blockein- und -ausgabebefehle dienen. Zum Glück können aber fast alle Hardware-Funktionen über Betriebssystem-Routinen angesprochen werden, so daß die direkte Programmierung der Ports der Ausnahmefall bleibt.

Sehr stark erweitert wurde die Leistungsfähigkeit der Rotierkommandos. Der 8080 erlaubte Rotationen nur mit dem Akku. Auch der Z80 kennt diese Befehle, zusätzlich kann er aber auch die anderen Register rotieren. Zudem gesellen sich zu den Rotationsbefehlen auch Schiebebefehle hinzu: SLA, SRA und SRL. Bei diesen werden die herausfallenden Bits nicht wieder auf der Gegenseite hereingebracht, sondern gehen unwiederbringlich verloren. Die Division und Multiplikation mit Zweier-

Potenzen wird dadurch wesentlich erleichtert. Man muß nicht mehr darauf achten, welche Bits am andern Ende des Registers wieder auftauchen.

Die BCD-Arithmetik wurde auch korrigiert. Der DAA-Befehl liefert bei Subtraktionen auf dem 8080 ein falsches Ergebnis. Beim Z80 wurde dieser Fehler glücklicherweise behoben. Zusätzlich gibt es noch RLD und RRD zur Arbeit mit den BCD-Zahlen (binäre codierte Dezimalzahlen).

Sie sehen also, die Z80-Kommandos sind so leistungsfähig, daß mit ihnen die Arbeit richtig Spaß macht. Verständlich, daß Programmierer nicht besonders »scharf« darauf sind, Programme 8080-kompatibel zu entwickeln: Die überwiegende Zahl der CP/M-Computer hat aber als Herz einen Z80. Und wenn Sie Ihre Software ausschließlich auf dem Schneider-CPC einsetzen wollen, können Sie natürlich alle Z80-Befehle benutzen. Leider sind die meisten CP/M-Assembler ausschließlich auf den 8080-Befehlssatz ausgerichtet und verarbeiten somit die Z80-Mnemonics überhaupt nicht.

Und richtig CP/M-kompatibel bleiben Ihre Programme nur, wenn Sie sich auf den 8080-Befehlssatz beschränken. In Tabelle 3 finden Sie die Befehle, die sowohl der Z80 als auch der 8080 besitzt. Tabelle 4 zeigt die Erweiterungen des Z80.

(Martin Kotulla/hg)

Tablle 3. Zwei Namen für die gleiche Arbeit – der gemeinsame Befehlssatz der CP/M-Prozessoren

Code	8080		Z80		Erklärung
CE	ACI	by	ADC	A,by	Addiere Byte mit Carry zum Akku
8F	ADC	A	ADC	A,A	Addiere Akku mit Carry zum Akku
88	ADC	B	ADC	A,B	Addiere B-Register mit Carry zum Akku
89	ADC	C	ADC	A,C	Addiere C-Register mit Carry zum Akku
8A	ADC	D	ADC	A,D	Addiere D-Register mit Carry zum Akku
8B	ADC	E	ADC	A,E	Addiere E-Register mit Carry zum Akku
8C	ADC	H	ADC	A,H	Addiere H-Register mit Carry zum Akku
8D	ADC	L	ADC	A,L	Addiere L-Register mit Carry zum Akku
8E	ADC	M	ADC	A,(HL)	Addiere (HL) zum Akku
87	ADD	A	ADD	A,A	Addiere Akku zum Akku
80	ADD	B	ADD	A,B	Addiere B-Register zum Akku
81	ADD	C	ADD	A,C	Addiere C-Register zum Akku
82	ADD	D	ADD	A,D	Addiere D-Register zum Akku
83	ADD	E	ADD	A,E	Addiere E-Register zum Akku
84	ADD	H	ADD	A,H	Addiere H-Register zum Akku
85	ADD	L	ADD	A,L	Addiere L-Register zum Akku
86	ADD	M	ADD	A,(HL)	Addiere (HL) zum Akku
C6	ADI	by	ADD	A,by	Addiere Byte zum Akku
A7	ANA	A	AND	A	Logisches Und mit Akku
A0	ANA	B	AND	B	Logisches Und mit B-Register
A1	ANA	C	AND	C	Logisches Und mit C-Register
A2	ANA	D	AND	D	Logisches Und mit D-Register
A3	ANA	E	AND	E	Logisches Und mit E-Register
A4	ANA	H	AND	H	Logisches Und mit H-Register
A5	ANA	L	AND	L	Logisches Und mit L-Register
A6	ANA	M	AND	(HL)	Logisches Und mit (HL)
E6	ANI	by	AND	by	Logisches Und mit Byte
CD	CALL	wo	CALL	wo	Unbedingter Unterprogrammaufruf
DC	CC	wo	CALL	C,wo	Unterprogrammaufruf, wenn Carry gesetzt
FC	CM	wo	CALL	M,wo	Unterprogrammaufruf, wenn Minus gesetzt
2F	CMA		CPL		Komplementiere Akku
3F	CMC		CCF		Komplementiere Carry-Flag
BF	CMP	A	CP	A	Vergleiche mit Akku
B8	CMP	B	CP	B	Vergleiche mit B-Register
B9	CMP	C	CP	C	Vergleiche mit C-Register
BA	CMP	D	CP	D	Vergleiche mit D-Register
BB	CMP	E	CP	E	Vergleiche mit E-Register
BC	CMP	H	CP	H	Vergleiche mit H-Register
BD	CMP	L	CP	L	Vergleiche mit L-Register
BE	CMP	M	CP	(HL)	Vergleiche mit (HL)
D4	CNC	wo	CALL	NC,wo	Unterprogrammaufruf, wenn Carry gelöscht
C4	CNZ	wo	CALL	NZ,wo	Unterprogrammaufruf wenn Zero gelöscht
F4	CP	wo	CALL	P,wo	Unterprogrammaufruf wenn Minus gelöscht
EC	CPE	wo	CALL	PE,wo	Unterprogrammaufruf wenn Parität gerade
FE	CPI	by	CP	by	Vergleiche mit Byte
E4	CPO	wo	CALL	PO,wo	Unterprogrammaufruf wenn Parität ungerade
CC	CZ	wo	CALL	Z,wo	Unterprogrammaufruf wenn Zero gesetzt
27	DAA		DAA		BCD-Dezimalkorrektur des Akkus
09	DAD	B	ADD	HL,BC	Addiere HL und BC, Ergebnis in HL
19	DAD	D	ADD	HL,DE	Addiere HL und DE, Ergebnis in HL
29	DAD	H	ADD	HL,HL	Addiere HL und HL, Ergebnis in HL
39	DAD	SP	ADD	HL,SP	Addiere HL und SP, Ergebnis in HL
3D	DCR	A	DEC	A	Dekrementiere Akku
05	DCR	B	DEC	B	Dekrementiere B-Register
0D	DCR	C	DEC	C	Dekrementiere C-Register
15	DCR	D	DEC	D	Dekrementiere D-Register
1D	DCR	E	DEC	E	Dekrementiere E-Register
25	DCR	H	DEC	H	Dekrementiere H-Register
2D	DCR	L	DEC	L	Dekrementiere L-Register
35	DCR	M	DEC	(HL)	Dekrementiere (HL)
0B	DCX	B	DEC	BC	Dekrementiere BC-Register
1B	DCX	D	DEC	DE	Dekrementiere DE-Register
2B	DCX	H	DEC	HL	Dekrementiere HL-Register
3B	DCX	SP	DEC	SP	Dekrementiere SP-Register
F3	DI		DI		Verbiete Interrupts
FB	EI		EI		Erlaube Interrupts
76	HLT		HALT		Stoppe Prozessor bis zum nächsten Interrupt
DB	IN	by	IN	A,by	Lese Port in den Akku
3C	INR	A	INC	A	Inkrementiere Akku
04	INR	B	INC	B	Inkrementiere B-Register
0C	INR	C	INC	C	Inkrementiere C-Register
14	INR	D	INC	D	Inkrementiere D-Register
1C	INR	E	INC	E	Inkrementiere E-Register
24	INR	H	INC	H	Inkrementiere H-Register
2C	INR	L	INC	L	Inkrementiere L-Register
34	INR	M	INC	(HL)	Inkrementiere (HL)
03	INX	B	INC	BC	Inkrementiere BC-Register
13	INX	D	INC	DE	Inkrementiere DE-Register
23	INX	H	INC	HL	Inkrementiere HL-Register
33	INX	SP	INC	SP	Inkrementiere SP-Register
DA	JC	wo	JP	C,wo	Sprung, wenn Carry-Flag gesetzt
FA	JM	wo	JP	M,wo	Sprung, wenn Minus-Flag gesetzt
C3	JMP	wo	JP	wo	Unbedingter Sprung
D2	JNC	wo	JP	NC,wo	Sprung, wenn Carry-Flag gelöscht
C2	JNZ	wo	JP	NZ,wo	Sprung, wenn Zero-Flag gelöscht
F2	JP	wo	JP	P,wo	Sprung, wenn Minus-Flag gelöscht
EA	JPE	wo	JP	PE,wo	Sprung, wenn Parität gerade
E2	JPO	wo	JP	PO,wo	Sprung, wenn Parität ungerade
CA	JZ	wo	JP	Z,wo	Sprung, wenn Zero-Flag gesetzt
3A	LDA	wo	LD	A,(wo)	Lade Akku mit Inhalt der Adresse
0A	LDAX	B	LD	A,(BC)	Lade Akku mit Inhalt der Adresse in BC
1A	LDAX	D	LD	A,(DE)	Lade Akku mit Inhalt der Adresse in DE
2A	LHLD	wo	LD	HL,(wo)	Lade HL mit 16-Bit-Inhalt der Adresse
01	LXI	B,wo	LD	BC,wo	Lade BC-Register unmittelbar
11	LXI	D,wo	LD	DE,wo	Lade DE-Register unmittelbar
21	LXI	H,wo	LD	HL,wo	Lade HL-Register unmittelbar
31	LXI	SP,wo	LD	SP,wo	Lade SP-Register unmittelbar
7F	MOV	A,A	LD	A,A	Lade Akku mit Akku
78	MOV	A,B	LD	A,B	Lade Akku mit B-Register
79	MOV	A,C	LD	A,C	Lade Akku mit C-Register
7A	MOV	A,D	LD	A,D	Lade Akku mit D-Register
7B	MOV	A,E	LD	A,E	Lade Akku mit E-Register
7C	A,H	MOV	LD	A,H	Lade Akku mit H-Register
7D	MOV	A,L	LD	A,L	Lade Akku mit L-Register
7E	MOV	A,M	LD	A,(HL)	Lade Akku mit (HL)
47	MOV	B,A	LD	B,A	Lade B-Register mit Akku
40	MOV	B,B	LD	B,B	Lade B-Register mit B-Register
41	MOV	B,C	LD	B,C	Lade B-Register mit C-Register
42	MOV	B,D	LD	B,D	Lade B-Register mit D-Register
43	MOV	B,E	LD	B,E	Lade B-Register mit E-Register
44	MOV	B,H	LD	B,H	Lade B-Register mit H-Register
45	MOV	B,L	LD	B,L	Lade B-Register mit L-Register
46	MOV	B,M	LD	B,(HL)	Lade B-Register mit HL
4F	MOV	C,A	LD	C,A	Lade C-Register mit Akku
48	MOV	C,B	LD	C,B	Lade C-Register mit B-Register
49	MOV	C,C	LD	C,C	Lade C-Register mit C-Register
4A	MOV	C,D	LD	C,D	Lade C-Register mit D-Register
4B	MOV	C,E	LD	C,E	Lade C-Register mit E-Register
4C	MOV	C,H	LD	C,H	Lade C-Register mit H-Register

Code	8080	Z80	Erklärung
4D	MOV C,L	LD C,L	Lade C-Register mit L-Register
4E	MOV C,M	LD C,(HL)	Lade C-Register mit (HL)
57	MOV D,A	LD D,A	Lade D-Register mit Akku
50	MOV D,B	LD D,B	Lade D-Register mit B-Register
51	MOV D,C	LD D,C	Lade D-Register mit C-Register
52	MOV D,D	LD D,D	Lade D-Register mit D-Register
53	MOV D,E	LD D,E	Lade D-Register mit E-Register
54	MOV D,H	LD D,H	Lade D-Register mit H-Register
55	MOV D,L	LD D,L	Lade D-Register mit L-Register
56	MOV D,M	LD D,(HL)	Lade D-Register mit (HL)
5F	MOV E,A	LD E,A	Lade E-Register mit Akku
58	MOV E,B	LD E,B	Lade E-Register mit B-Register
59	MOV E,C	LD E,C	Lade E-Register mit C-Register
5A	MOV E,D	LD E,D	Lade E-Register mit D-Register
5B	MOV E,E	LD E,E	Lade E-Register mit E-Register
5C	MOV E,H	LD E,H	Lade E-Register mit H-Register
5D	MOV E,L	LD E,L	Lade E-Register mit L-Register
5E	MOV E,M	LD E,(HL)	Lade E-Register mit (HL)
67	MOV H,A	LD H,A	Lade H-Register mit Akku
60	MOV H,B	LD H,B	Lade H-Register mit B-Register
61	MOV H,C	LD H,C	Lade H-Register mit C-Register
62	MOV H,D	LD H,D	Lade H-Register mit D-Register
63	MOV H,E	LD H,E	Lade H-Register mit E-Register
64	MOV H,H	LD H,H	Lade H-Register mit H-Register
65	MOV H,L	LD H,L	Lade H-Register mit L-Register
66	MOV H,M	LD H,(HL)	Lade H-Register mit (HL)
6F	MOV L,A	LD L,A	Lade L-Register mit Akku
68	MOV L,B	LD L,B	Lade L-Register mit B-Register
69	MOV L,C	LD L,C	Lade L-Register mit C-Register
6A	MOV L,D	LD L,D	Lade L-Register mit D-Register
6B	MOV L,E	LD L,E	Lade L-Register mit E-Register
6C	MOV L,H	LD L,H	Lade L-Register mit H-Register
6D	MOV L,L	LD L,L	Lade L-Register mit L-Register
6E	MOV L,M	LD L,(HL)	Lade L-Register mit (HL)
77	MOV M,A	LD (HL),A	Lade (HL) mit Akku
70	MOV M,B	LD (HL),B	Lade (HL) mit B-Register
71	MOV M,C	LD (HL),C	Lade (HL) mit C-Register
72	MOV M,D	LD (HL),D	Lade (HL) mit D-Register
73	MOV M,E	LD (HL),E	Lade (HL) mit E-Register
74	MOV M,H	LD (HL),H	Lade (HL) mit H-Register
75	MOV M,L	LD (HL),L	Lade (HL) mit L-Register
3E	MVI A,by	LD A,by	Lade Akku mit Byte
06	MVI B,by	LD B,by	Lade B-Register mit Byte
0E	MVI C,by	LD C,by	Lade C-Register mit Byte
16	MVI D,by	LD D,by	Lade D-Register mit Byte
1E	MVI E,by	LD E,by	Lade E-Register mit Byte
26	MVI H,by	LD H,by	Lade H-Register mit Byte
2E	MVI L,by	LD L,by	Lade L-Register mit Byte
36	MVI M,by	LD (HL),by	Lade (HL) mit Byte
00	NOP	NOP	Tut nichts, wartet nur kurz
B7	ORA A	OR A	Logisches Oder mit Akku
B0	ORA B	OR B	Logisches Oder mit B-Register
B1	ORA C	OR C	Logisches Oder mit C-Register
B2	ORA D	OR D	Logisches Oder mit D-Register
B3	ORA E	OR E	Logisches Oder mit E-Register
B4	ORA H	OR H	Logisches Oder mit H-Register
B5	ORA L	OR L	Logisches Oder mit L-Register
B6	ORA M	OR (HL)	Logisches Oder mit (HL)
F6	ORI by	OR by	Logisches Oder mit Byte
D3	OUT by	OUT (by),A	Akku auf Port ausgeben
E9	PCHL	JP (HL)	Springe unbedingt zur Adresse in HL
C1	POP B	POP BC	Hole BC-Register vom Stack
D1	POP D	POP DE	Hole DE-Register vom Stack
E1	POP H	POP HL	Hole HL-Register vom Stack
F1	POP PSW	POP AF	Hole Akku und Flag-Register vom Stack
C5	PUSHB	PUSH BC	Schiebe BC-Register auf den Stack
D5	PUSHD	PUSH DE	Schiebe DE-Register auf den Stack
E5	PUSHH	PUSH H	Schiebe HL-Register auf den Stack
F5	PUSHPSW	PUSH AF	Schiebe Akku und Flag-Register auf den Stack
17	RAL	RLA	Rotiere Akku nach links über Carry-Flag
1F	RAR	RRA	Rotiere Akku nach rechts über Carry-Flag
D8	RC	RET C	Rücksprung, wenn Carry-Flag gesetzt
C9	RET	RET	Unbedingter Rücksprung
07	RLC	RLCA	Rotiere Akku nach links ohne Carry
F8	RM	RET M	Rücksprung, wenn Minus-Flag gesetzt
D0	RNC	RET NC	Rücksprung, wenn Carry-Flag gelöscht
C0	RNZ	RET NZ	Rücksprung, wenn Zero-Flag gelöscht
F0	RP	RET P	Rücksprung, wenn Minus-Flag gelöscht
E8	RPE	RET PE	Rücksprung, wenn Parität gerade
E0	RPO	RET PO	Rücksprung, wenn Parität ungerade
0F	RRC	RRCA	Rotiere Akku nach rechts ohne Carry
C7	RST 0	RST0 0000H	Restart zur Adresse 00H
CF	RST 1	RST1 0008H	Restart zur Adresse 08H
D7	RST 2	RST2 0010H	Restart zur Adresse 10H
DF	RST 3	RST3 0018H	Restart zur Adresse 18H
E7	RST 4	RST4 0020H	Restart zur Adresse 20H
EF	RST 5	RST5 0028H	Restart zur Adresse 28H
F7	RST 6	RST6 0030H	Restart zur Adresse 30H
FF	RST 7	RST7 0038H	Restart zur Adresse 38H
C8	RZ	RET Z	Rücksprung, wenn Zero-Flag gesetzt
9F	SBB A	SBC A,A	Subtrahiere Akku mit Carry
98	SBB B	SBC A,B	Subtrahiere B-Register mit Carry
99	SBB C	SBC A,C	Subtrahiere C-Register mit Carry
9A	SBB D	SBC A,D	Subtrahiere D-Register mit Carry
9B	SBB E	SBC A,E	Subtrahiere E-Register mit Carry
9C	SBB H	SBC A,H	Subtrahiere H-Register mit Carry
9D	SBB L	SBC A,L	Subtrahiere L-Register mit Carry
9E	SBB M	SBC A,(HL)	Subtrahiere (HL) mit Carry
DE	SBI by	SBC A,by	Subtrahiere Byte mit Carry
22	SHLD wo	LD (wo),HL	Lade HL-Register an die Adresse
F9	SPHL	LD SP,HL	Lade SP-Register mit HL-Register
32	STA wo	LD (wo),A	Lade Akku an die Adresse
02	STAX B	LD (BC),A	Lade Akku an die Adresse in BC
12	STAX D	LD (DE),A	Lade Akku an die Adresse in DE
37	STC	SCF	Setze Carry-Flag
97	SUB A	SUB A	Subtrahiere Akku
90	SUB B	SUB B	Subtrahiere B-Register
91	SUB C	SUB C	Subtrahiere C-Register
92	SUB D	SUB D	Subtrahiere D-Register
93	SUB E	SUB E	Subtrahiere E-Register
94	SUB H	SUB H	Subtrahiere H-Register
95	SUB L	SUB L	Subtrahiere L-Register
96	SUB M	SUB (HL)	Subtrahiere (HL)
D6	SUI by	SUB by	Subtrahiere Byte
EB	XCHG	EX DE,HL	Tausche HL-Register und DE-Register
AF	XRA A	XOR A	Logisches Exklusiv-Oder mit Akku
A8	XRA B	XOR B	Logisches Exklusiv-Oder mit B-Register
A9	XRA C	XOR C	Logisches Exklusiv-Oder mit C-Register
AA	XRA D	XOR D	Logisches Exklusiv-Oder mit D-Register
AB	XRA E	XOR E	Logisches Exklusiv-Oder mit E-Register
AC	XRA H	XOR H	Logisches Exklusiv-Oder mit H-Register
AD	XRA L	XOR L	Logisches Exklusiv-Oder mit L-Register
AE	XRA M	XOR (HL)	Logisches Exklusiv-Oder mit (HL)
EE	XRI by	XOR by	Logisches Exklusiv-Oder mit Byte
E3	XTHL	EX (SP),HL	Tausche (SP) mit HL-Register

by = Byte (8Bit)
wo = Wort (16 Bit)
di = Distanz (8 Bit)
(HL) = Adresse, auf die das HL-Register zeigt

Tabelle 3. Zwei Namen für die gleiche Arbeit – der gemeinsame Befehlssatz der CP/M-Prozessoren (Schluß)

Tabelle 4. Z80-Befehle, die der 8080 nicht kennt

di = Distanz (7 Bit und Vorzeichen)
by = 8-Bit-Zahl (Byte)
wo = 16-Bit-Zahl (Word)

Code	Befehl	Operand	Erklärung
DD 8E di	ADC	A,(IX+di)	Addiere mit Wert an Adresse IX+di und Carry
FD 8E di	ADC	A,(IY+di)	Addiere mit Wert an Adresse IY+di und Carry
ED 4A	ADC	HL,BC	Addiere HL und BC mit Carry, Ergebnis in HL
ED 5A	ADC	HL,DE	Addiere HL und DE mit Carry, Ergebnis in HL
ED 6A	ADC	HL,HL	Addiere HL und HL mit Carry, Ergebnis in HL
ED 7A	ADC	HL,SP	Addiere HL und SP mit Carry, Ergebnis in HL
DD 86 di	ADD	A,(IX+di)	Addiere mit Wert an Adresse IX+di
FD 86 di	ADD	A,(IY+di)	Addiere mit Wert an Adresse IY+di

Tabelle 4. Z80-Befehle, die der 8080 nicht kennt (Fortsetzung)

DD 09	ADD	IX,BC	Addiere IX und BC, Ergebnis in IX
DD 19	ADD	IX,DE	Addiere IX und DE, Ergebnis in IX
DD 29	ADD	IX,IX	Addiere IX und IX, Ergebnis in IX
DD 39	ADD	IX,SP	Addiere IX und SP, Ergebnis in IX
FD 09	ADD	IY,BC	Addiere IY und BC, Ergebnis in IY
FD 19	ADD	IY,DE	Addiere IY und DE, Ergebnis in IY
FD 29	ADD	IY,IY	Addiere IY und IY, Ergebnis in IY
FD 39	ADD	IY,SP	Addiere IY und SP, Ergebnis in IY
DD A6 di	AND	(IX+di)	Logisches Und mit (IX+di)
FD A6 di	AND	(IY+di)	Logisches Und mit (IY+di)
CB 46	BIT	0, (HL)	Teste Bit 0 in (HL)
DD CB di 46	BIT	0, (IX+di)	Teste Bit 0 in (IX+di)
FD CB di 46	BIT	0, (IY+di)	Teste Bit 0 in (IY+di)
CB 47	BIT	0,A	Teste Bit 0 im Akku
CB 40	BIT	0,B	Teste Bit 0 im B-Register
CB 41	BIT	0,C	Teste Bit 0 im C-Register
CB 42	BIT	0,D	Teste Bit 0 im D-Register
CB 43	BIT	0,E	Teste Bit 0 im E-Register
CB 44	BIT	0,H	Teste Bit 0 im H-Register
CB 45	BIT	0,L	Teste Bit 0 im L-Register
CB 4E	BIT	1, (HL)	Teste Bit 0 in (HL)
DD CB di 4E	BIT	1, (IX+di)	Teste Bit 1 in (IX+di)
FD CB di 4E	BIT	1, (IY+di)	Teste Bit 1 in (IY+di)
CB 4F	BIT	1,A	Teste Bit 1 im Akku
CB 48	BIT	1,B	Teste Bit 1 im B-Register
CB 49	BIT	1,C	Teste Bit 1 im C-Register
CB 4A	BIT	1,D	Teste Bit 1 im D-Register
CB 4B	BIT	1,E	Teste Bit 1 im E-Register
CB 4C	BIT	1,H	Teste Bit 1 im H-Register
CB 4D	BIT	1,L	Teste Bit 1 im L-Register
CB 56	BIT	2, (HL)	Teste Bit 2 in (HL)
DD CB di 56	BIT	2, (IX+di)	Teste Bit 2 in (IX+di)
FD CB di 56	BIT	2, (IY+di)	Teste Bit 2 in (IY+di)
CB 57	BIT	2,A	Teste Bit 2 im Akku
CB 50	BIT	2,B	Teste Bit 2 im B-Register
CB 51	BIT	2,C	Teste Bit 2 im C-Register
CB 52	BIT	2,D	Teste Bit 2 im D-Register
CB 53	BIT	2,E	Teste Bit 2 im E-Register
CB 54	BIT	2,H	Teste Bit 2 im H-Register
CB 55	BIT	2,L	Teste Bit 2 im L-Register
CB 5E	BIT	3, (HL)	Teste Bit 3 in (HL)
DD CB di 5E	BIT	3, (IX+di)	Teste Bit 3 in (IX+di)
FD CB di 5E	BIT	3, (IY+di)	Teste Bit 3 in (IY+di)
CB 5F	BIT	3,A	Teste Bit 3 im Akku
CB 58	BIT	3,B	Teste Bit 3 im B-Register
CB 59	BIT	3,C	Teste Bit 3 im C-Register
CB 5A	BIT	3,D	Teste Bit 3 im D-Register
CB 5B	BIT	3,E	Teste Bit 3 im E-Register
CB 5C	BIT	3,H	Teste Bit 3 im H-Register
CB 5D	BIT	3,L	Teste Bit 3 im L-Register
CB 66	BIT	4, (HL)	Teste Bit 4 in (HL)
DD CB di 66	BIT	4, (IX+di)	Teste Bit 4 in (IX+di)
FD CB di 66	BIT	4, (IY+di)	Teste Bit 4 in (IY+di)
CB 67	BIT	4,A	Teste Bit 4 im Akku
CB 60	BIT	4,B	Teste Bit 4 im B-Register
CB 61	BIT	4,C	Teste Bit 4 im C-Register
CB 62	BIT	4,D	Teste Bit 4 im D-Register
CB 63	BIT	4,E	Teste Bit 4 im E-Register
CB 64	BIT	4,H	Teste Bit 4 im H-Register
CB 65	BIT	4,L	Teste Bit 4 im L-Register
CB 6E	BIT	5, (HL)	Teste Bit 5 in (HL)
DD CB di 6E	BIT	5, (IX+di)	Teste Bit 5 in (IX+di)
FD CB di 6E	BIT	5, (IY+di)	Teste Bit 5 in (IY+di)
CB 6F	BIT	5,A	Teste Bit 5 im Akku
CB 68	BIT	5,B	Teste Bit 5 im B-Register
CB 69	BIT	5,C	Teste Bit 5 im C-Register
CB 6A	BIT	5,D	Teste Bit 5 im D-Register
CB 6B	BIT	5,E	Teste Bit 5 im E-Register
CB 6C	BIT	5,H	Teste Bit 5 im H-Register
CB 6D	BIT	5,L	Teste Bit 5 im L-Register
CB 76	BIT	6, (HL)	Teste Bit 6 in (HL)
DD CB di 76	BIT	6, (IX+di)	Teste Bit 6 in (IX+di)
FD CB di 76	BIT	6, (IY+di)	Teste Bit 6 in (IY+di)
CB 77	BIT	6,A	Teste Bit 6 im Akku
CB 70	BIT	6,B	Teste Bit 6 im B-Register
CB 71	BIT	6,C	Teste Bit 6 im C-Register
CB 72	BIT	6,D	Teste Bit 6 im D-Register
CB 73	BIT	6,E	Teste Bit 6 im E-Register
CB 74	BIT	6,H	Teste Bit 6 im H-Register
CB 75	BIT	6,L	Teste Bit 6 im L-Register
CB 7E	BIT	7, (HL)	Teste Bit 7 in (HL)
DD CB di 7E	BIT	7, (IX+di)	Teste Bit 7 in (IX+di)
FD CB di 7E	BIT	7, (IY+di)	Teste Bit 7 in (IY+di)
CB 7F	BIT	7,A	Teste Bit 7 im Akku
CB 78	BIT	7,B	Teste Bit 7 im B-Register
CB 79	BIT	7,C	Teste Bit 7 im C-Register
CB 7A	BIT	7,D	Teste Bit 7 im D-Register
CB 7B	BIT	7,E	Teste Bit 7 im E-Register
CB 7C	BIT	7,H	Teste Bit 7 im H-Register
CB 7D	BIT	7,L	Teste Bit 7 im L-Register
DD BE di	CP	(IX+di)	Vergleiche mit Wert in (IX+di)
FD BE di	CP	(IY+di)	Vergleiche mit Wert in (IY+di)
ED A9	CPD		Blocksuchbefehl ohne Wiederholung
ED B9	CPDR		Blocksuchbefehl mit Wiederholung
ED A1	CPI		Blocksuchbefehl ohne Wiederholung
ED B1	CPIR		Blocksuchbefehl mit Wiederholung
DD 35 di	DEC	(IX+di)	Dekrementiere Wert in (IX+di)
FD 35 di	DEC	(IY+di)	Dekrementiere Wert in (IY+di)
DD 2B	DEC	IX	Dekrementiere IX-Register
FD 2B	DEC	IY	Dekrementiere IY-Register
10 di	DJNZ	di	Sprung in die Schleife, bis B=0
DD E3	EX	(SP),IX	Tausche Stackspitze mit IX-Register
FD E3	EX	(SP),IY	Tausche Stackspitze mit IY-Register
08	EX	AF,AF	Tausche Akku & Flags mit Zweitregistern
D9	EXX		Tausche BC, DE und HL mit Zweitregistern
ED 46	IM	0	Interrupt-Modus 0
ED 56	IM	1	Interrupt-Modus 1
ED 5E	IM	2	Interrupt-Modus 2
ED 78	IN	A,(C)	Port-Einlesebefehl in den Akku
ED 40	IN	B,(C)	Port-Einlesebefehl ins B-Register
ED 48	IN	C,(C)	Port-Einlesebefehl ins C-Register
ED 50	IN	D,(C)	Port-Einlesebefehl ins D-Register
ED 58	IN	E,(C)	Port-Einlesebefehl ins E-Register
ED 60	IN	H,(C)	Port-Einlesebefehl ins H-Register
ED 68	IN	L,(C)	Port-Einlesebefehl ins L-Register
DD 34 di	INC	(IX+di)	Inkrementiere Wert in (IX+di)
FD 34 di	INC	(IY+di)	Inkrementiere Wert in (IY+di)
DD 23	INC	IX	Inkrementiere IX-Register
FD 23	INC	IY	Inkrementiere IY-Register
ED AA	IND		Blockeingabe ohne Wiederholung
ED BA	INDR		Blockeingabe mit Wiederholung
ED A2	INI		Blockeingabe ohne Wiederholung
ED B2	INIR		Blockeingabe mit Wiederholung
DD E9	JP	(IX)	Sprung zur Adresse in IX
FD E9	JP	(IY)	Sprung zur Adresse in IY
38 di	JR	C,di	Relativsprung, wenn Carry gesetzt
18 di	JR	di	Unbedingter Relativsprung
30	JR	NC,di	Relativsprung, wenn Carry gelöscht
20 di	JR	NZ,di	Relativsprung, wenn Zero gelöscht
28	JR	Z,di	Relativsprung, wenn Zero gesetzt
DD 77 di	LD	(IX+di),A	Lade Akku nach (IX+di)
DD 70 di	LD	(IX+di),B	Lade B-Register nach (IX+di)
DD 71 di	LD	(IX+di),C	Lade C-Register nach IX+di)
DD 72 di	LD	(IX+di),D	Lade D-Register nach (IX+di)
DD 73 di	LD	(IX+di),E	Lade E-Register nach (IX+di)
DD 74 di	LD	(IX+di),H	Lade H-Register nach (IX+di)
DD 75 di	LD	(IX+di),L	Lade L-Register nach (IX+di)
DD 36 diby	LD	(IX+di),by	Lade Byte nach (IX+di)
FD 77 di	LD	(IY+di),A	Lade Akku nach (IY+di)
FD 70 di	LD	(IY+di),B	Lade B-Register nach (IY+di)
FD 71 di	LD	(IY+di),C	Lade C-Register nach (IY+di)
FD 72 di	LD	(IY+di),D	Lade D-Register nach (IY+di)
FD 73 di	LD	(IY+di),E	Lade E-Register nach (IY+di)
FD 74 di	LD	(IY+di),H	Lade H-Register nach (IY+di)
FD 75 di	LD	(IY+di),L	Lade L-Register nach (IY+di)
FD 36 diby	LD	(IY+di),by	Lade Byte nach (IY+di)
ED 43 wo	LD	(wo),BC	Lade BC nach Adresse
ED 53 wo	LD	(wo),DE	Lade DE nach Adresse
DD 22 wo	LD	(wo),IX	Lade IX nach Adresse
FD 22 wo	LD	(wo),IY	Lade IY nach Adresse
ED 73 wo	LD	(wo),SP	Lade SP nach Adresse
DD 7E di	LD	A,(IX+di)	Lade Akku aus (IX+di)
FD 7E di	LD	A,(IY+di)	Lade Akku aus (IY+di)
ED 57	LD	A,I	Lade Akku mit I-Register
ED 5F	LD	A,R	Lade Akku mit R-Register
DD 46 di	LD	B,(IX+di)	Lade B-Register aus (IX+di)
FD 46 di	LD	B,(IY+di)	Lade B-Register aus (IY+di)
ED 4B wo	LD	BC,(wo)	Lade BC-Register aus Adresse
DD 4E di	LD	C,(IX+di)	Lade C-Register aus (IX+di)
FD 4E di	LD	C,(IY+di)	Lade C-Register aus (IY+di)
DD 56 di	LD	D,(IX+di)	Lade D-Register aus (IX+di)
FD 56 di	LD	D,(IY+di)	Lade D-Register aus (IY+di)
ED 5B wo	LD	DE,(wo)	Lade DE-Register aus Adresse
DD 5E di	LD	E,(IX+di)	Lade E-Register aus (IX+di)



FD 5E di	LD	E,(IY+di)	Lade E-Register aus (IY+di)
DD 66 di	LD	H,(IX+di)	Lade H-Register aus (IX+di)
FD 66 di	LD	L,(IY+di)	Lade L-Register aus (IY+di)
ED 47	LD	I,A	Lade I-Register mit Akku
DD 2A wo	LD	IX,(wo)	Lade IX-Register mit Wert an Adresse
DD 21 wo	LD	IX,wo	Lade IX-Register mit 16-Bit-Wert
FD 2A wo	LD	IY,(wo)	Lade IY-Register mit Wert an Adresse
FD 21 wo	LD	IY,wo	Lade IY-Register mit 16-Bit-Wert
DD 6E di	LD	L,(IX+di)	Lade L-Register aus (IX+di)
FD 6E di	LD	L,(IY+di)	Lade L-Register aus (IY+di)
ED 4F	LD	R,A	Lade R-Register mit Akku
ED 7B wo	LD	SP,(wo)	Lade SP-Register aus Adresse
DD F9	LD	SP,IX	Lade SP-Register mit IX-Register
FD F9	LD	SP,IY	Lade SP-Register mit IY-Register
ED A8	LDD		Block kopieren ohne Wiederholung
ED B8	LDDR		Block kopieren mit Wiederholung
ED A0	LDI		Block kopieren ohne Wiederholung
ED B0	LDIR		Block kopieren mit Wiederholung
ED 44	NEG		Akku = -Akku, Zweier-Komplement
DD B6 di	OR	(IX+di)	Logisches Oder mit (IX+di)
FD B6 di	OR	(IY+di)	Logisches Oder mit (IY+di)
ED BB	OTDR		Blockausgabebefehl mit Wiederholung
ED B3	OTIR		Blockausgabebefehl mit Wiederholung
ED 79	OUT	(C),A	Portausgabe des Akkus
ED 41	OUT	(C),B	Portausgabe des B-Registers
ED 49	OUT	(C),C	Portausgabe des C-Registers
ED 51	OUT	(C),D	Portausgabe des D-Registers
ED 59	OUT	(C),E	Portausgabe des E-Registers
ED 61	OUT	(C),H	Portausgabe des H-Registers
ED 69	OUT	(C),L	Portausgabe des L-Registers
ED AB	OUTD		Blockausgabebefehl ohne Wiederholung
ED A3	OUTI		Blockausgabebefehl ohne Wiederholung
DD E1	POP	IX	IX-Register vom Stack holen
FD E1	POP	IY	IY-Register vom Stack holen
DD E5	PUSH	IX	IX-Register auf den Stack schieben
FD E5	PUSH	IY	IY-Register auf den Stack schieben
CB 86	RES	0, (HL)	Lösche Bit 0 in (HL)
DD CB di 86	RES	0, (IX+di)	Lösche Bit 0 in (IX+di)
FD CB di 86	RES	0, (IY+di)	Lösche Bit 0 in (IY+di)
CB 87	RES	0,A	Lösche Bit 0 im Akku
CB 80	RES	0,B	Lösche Bit 0 im B-Register
CB 81	RES	0,C	Lösche Bit 0 im C-Register
CB 82	RES	0,D	Lösche Bit 0 im D-Register
CB 83	RES	0,E	Lösche Bit 0 im E-Register
CB 84	RES	0,H	Lösche Bit 0 im H-Register
CB 85	RES	0,L	Lösche Bit 0 im L-Register
CB 8E	RES	1,(HL)	Lösche Bit 1 in (HL)
DD CB di 8E	RES	1,(IX+di)	Lösche Bit 1 in (IX+di)
FD CB di 8E	RES	1,(IY+di)	Lösche Bit 1 in (IY+di)
CB 8F	RES	1,A	Lösche Bit 1 im Akku
CB 88	RES	1,B	Lösche Bit 1 im B-Register
CB 89	RES	1,C	Lösche Bit 1 im C-Register
CB 8A	RES	1,D	Lösche Bit 1 im D-Register
CB 8B	RES	1,E	Lösche Bit 1 im E-Register
CB 8C	RES	1,H	Lösche Bit 1 im H-Register
CB 8D	RES	1,L	Lösche Bit 1 im L-Register
CB 96	RES	2,(HL)	Lösche Bit 2 in (HL)
DD CB di 96	RES	2,(IX+di)	Lösche Bit 2 in (IX+di)
FD CB di 96	RES	2,(IY+di)	Lösche Bit 2 in (IY+di)
CB 97	RES	2,A	Lösche Bit 2 im Akku
CB 90	RES	2,B	Lösche Bit 2 im B-Register
CB 91	RES	2,C	Lösche Bit 2 im C-Register
CB 92	RES	2,D	Lösche Bit 2 im D-Register
CB 93	RES	2,E	Lösche Bit 2 im E-Register
CB 94	RES	2,H	Lösche Bit 2 im H-Register
CB 95	RES	2,L	Lösche Bit 2 im L-Register
CB 9E	RES	3,(HL)	Lösche Bit 3 in (HL)
DD CB di 9E	RES	3,(IX+di)	Lösche Bit 1 in (IX+di)
FD CB di 9E	RES	3,(IY+di)	Lösche Bit 1 in (IY+di)
CB 9F	RES	3,A	Lösche Bit 3 im Akku
CB 98	RES	3,B	Lösche Bit 3 im B-Register
CB 99	RES	3,C	Lösche Bit 3 im C-Register
CB 9A	RES	3,D	Lösche Bit 3 im D-Register
CB 9B	RES	3,E	Lösche Bit 3 im E-Register
CB 9C	RES	3,H	Lösche Bit 3 im H-Register
CB 9D	RES	3,L	Lösche Bit 3 im L-Register
CB A6	RES	4,(HL)	Lösche Bit 4 in (HL)
DD CB di A6	RES	4,(IX+di)	Lösche Bit 4 in (IX+di)
FD CB di A6	RES	4,(IY+di)	Lösche Bit 4 in (IY+di)
CB A7	RES	4,A	Lösche Bit 4 im Akku
CB A0	RES	4,B	Lösche Bit 4 im B-Register
CB A1	RES	4,C	Lösche Bit 4 im C-Register
CB A2	RES	4,D	Lösche Bit 4 im D-Register
CB A3	RES	4,E	Lösche Bit 4 im E-Register
CB A4	RES	4,H	Lösche Bit 4 im H-Register
CB A5	RES	4,L	Lösche Bit 4 im L-Register
CB AE	RES	5,(HL)	Lösche Bit 5 in (HL)
DD CB di AE	RES	5,(IX+di)	Lösche Bit 5 in (IX+di)
FD CB di AE	RES	5,(IY+di)	Lösche Bit 5 in (IY+di)
CB AF	RES	5,A	Lösche Bit 5 im Akku
CB A8	RES	5,B	Lösche Bit 5 im B-Register
CB A9	RES	5,C	Lösche Bit 5 im C-Register
CB AA	RES	5,D	Lösche Bit 5 im D-Register
CB AB	RES	5,E	Lösche Bit 5 im E-Register
CB AC	RES	5,H	Lösche Bit 5 im H-Register
CB AD	RES	5,L	Lösche Bit 5 im L-Register
CB B6	RES	6,(HL)	Lösche Bit 6 in (HL)
DD CB di B6	RES	6,(IX+di)	Lösche Bit 6 in (IX+di)
FD CB di B6	RES	6,(IY+di)	Lösche Bit 6 in (IY+di)
CB B7	RES	6,A	Lösche Bit 6 im Akku
CB B0	RES	6,B	Lösche Bit 6 im B-Register
CB B1	RES	6,C	Lösche Bit 6 im C-Register
CB B2	RES	6,D	Lösche Bit 6 im D-Register
CB B3	RES	6,E	Lösche Bit 6 im E-Register
CB B4	RES	6,H	Lösche Bit 6 im H-Register
CB B5	RES	6,L	Lösche Bit 6 im L-Register
CB BE	RES	7,(HL)	Lösche Bit 7 in (HL)
DD CB di BE	RES	7,(IX+di)	Lösche Bit 7 in (IX+di)
FD CB di BE	RES	7,(IY+di)	Lösche Bit 7 in (IY+di)
CB BF	RES	7,A	Lösche Bit 7 im Akku
CB B8	RES	7,B	Lösche Bit 7 im B-Register
CB B9	RES	7,C	Lösche Bit 7 im C-Register
CB BA	RES	7,D	Lösche Bit 7 im D-Register
CB BB	RES	7,E	Lösche Bit 7 im E-Register
CB BC	RES	7,H	Lösche Bit 7 im H-Register
CB BD	RES	7,L	Lösche Bit 7 im L-Register
ED 4D	RETI		Rücksprung von Interrupt-Routine
ED 45	RETN		Rücksprung von NMI-Interrupt
CB 16	RL	(HL)	Linksrotation von (HL) durchs Carry
DD CB di 16	RL	(IX+di)	Linksrotation von (IX+di) durchs Carry
FD CB di 16	RL	(IY+di)	Linksrotation von (IY+di) durchs Carry
CB 17	RL	A	Linksrotation des Akkus durchs Carry
CB 10	RL	B	Linksrotation des B-Registers durchs Carry
CB 11	RL	C	Linksrotation des C-Registers durchs Carry
CB 12	RL	D	Linksrotation des D-Registers durchs Carry
CB 13	RL	E	Linksrotation des E-Registers durchs Carry
CB 14	RL	H	Linksrotation des H-Registers durchs Carry
CB 15	RL	L	Linksrotation des L-Registers durchs Carry
CB 06	RLC	(HL)	Linksrotation von (HL) ohne Carry
DD CB di 06	RLC	(IX+di)	Linksrotation von (IX+di) ohne Carry
FD CB di 06	RLC	(IY+di)	Linksrotation von (IY+di) ohne Carry
CB 07	RLC	A	Linksrotation des Akkus ohne Carry
CB 00	RLC	B	Linksrotation des B-Registers ohne Carry
CB 01	RLC	C	Linksrotation des C-Registers ohne Carry
CB 02	RLC	D	Linksrotation des D-Registers ohne Carry
CB 03	RLC	E	Linksrotation des E-Registers ohne Carry
CB 04	RLC	H	Linksrotation des H-Registers ohne Carry
CB 05	RLC	L	Linksrotation des L-Registers ohne Carry
ED 6F	RLD		BCD-Zahlenrotation mit (HL) und Akku
CB 1E	RR	(HL)	Rechtsrotation von (HL) durchs Carry
DD CB di 1E	RR	(IX+di)	Rechtsrotation von (IX+di) durchs Carry
FD CB di 1E	RR	(IY+di)	Rechtsrotation von (IY+di) durchs Carry
CB 1F	RR	A	Rechtsrotation des Akkus durchs Carry
CB 18	RR	B	Rechtsrotation des B-Registers durchs Carry
CB 19	RR	C	Rechtsrotation des C-Registers durchs Carry
CB 1A	RR	D	Rechtsrotation des D-Registers durchs Carry

Tabelle 4. Z80-Befehle, die der 8080 nicht kennt (Schluß)

Code	Befehl	Operand	Beschreibung
CB 1B	RR	E	Rechtsrotation des E-Registers durchs Carry
CB 1C	RR	H	Rechtsrotation des H-Registers durchs Carry
CB 1D	RR	L	Rechtsrotation des L-Registers durchs Carry
CB 0E	RRC	(HL)	Rechtsrotation von (HL) ohne Carry
DD CB di 0E	RRC	(IX+di)	Rechtsrotation von (IX+di) ohne Carry
FD CB di 0E	RRC	(IY+di)	Rechtsrotation von (IY+di) ohne Carry
CB 0F	RRC	A	Rechtsrotation des Akkus ohne Carry
CB 08	RRC	B	Rechtsrotation des B-Registers ohne Carry
CB 09	RRC	C	Rechtsrotation des C-Registers ohne Carry
CB 0A	RRC	D	Rechtsrotation des D-Registers ohne Carry
CB 0B	RRC	E	Rechtsrotation des E-Registers ohne Carry
CB 0C	RRC	H	Rechtsrotation des H-Registers ohne Carry
CB 0D	RRC	L	Rechtsrotation des L-Registers ohne Carry
ED 67	RRD		BCD-Zahlenrotation mit (HL) und Akku
DD 9E di	SBC	A,(IX+di)	Subtraktion mit (IX+di) und Carry
FD 9E di	SBC	A,(IY+di)	Subtraktion mit (IY+di) und Carry
ED 42	SBC	HL,BC	HL:=HL-BC, Carry
ED 52	SBC	HL,DE	HL:=HL-DE, Carry
ED 62	SBC	HL,HL	HL:=HL-HL, Carry
ED 72	SBC	HL,SP	HL:=HL-SP, Carry
CB C6	SET	0,(HL)	Setze Bit 0 in (HL)
DD CB di C6	SET	0,(IX+di)	Setze Bit 0 in (IX+di)
FD CB di C6	SET	0,(IY+di)	Setze Bit 0 in (IY+di)
CB C7	SET	0,A	Setze Bit 0 im Akku
CB C0	SET	0,B	Setze Bit 0 im B-Register
CB C1	SET	0,C	Setze Bit 0 im C-Register
CB C2	SET	0,D	Setze Bit 0 im D-Register
CB C3	SET	0,E	Setze Bit 0 im E-Register
CB C4	SET	0,H	Setze Bit 0 im H-Register
CB C5	SET	0,L	Setze Bit 0 im L-Register
CB CE	SET	1,(HL)	Setze Bit 1 in (HL)
DD CB di CE	SET	1,(IX+di)	Setze Bit 1 in (IX+di)
FD CB di CE	SET	1,(IY+di)	Setze Bit 1 in (IY+di)
CB CF	SET	1,A	Setze Bit 1 im Akku
CB C8	SET	1,B	Setze Bit 1 im B-Register
CB C9	SET	1,C	Setze Bit 1 im C-Register
CB CA	SET	1,D	Setze Bit 1 im D-Register
CB CB	SET	1,E	Setze Bit 1 im E-Register
CB CC	SET	1,H	Setze Bit 1 im H-Register
CB CD	SET	1,L	Setze Bit 1 im L-Register
CB D6	SET	2,(HL)	Setze Bit 2 in (HL)
DD CB di D6	SET	2,(IX+di)	Setze Bit 2 in (IX+di)
FD CB di D6	SET	2,(IY+di)	Setze Bit 2 in (IY+di)
CB D7	SET	2,A	Setze Bit 2 im Akku
CB D0	SET	2,B	Setze Bit 2 im B-Register
CB D1	SET	2,C	Setze Bit 2 im C-Register
CB D2	SET	2,D	Setze Bit 2 im D-Register
CB D3	SET	2,E	Setze Bit 2 im E-Register
CB D4	SET	2,H	Setze Bit 2 im H-Register
CB D5	SET	2,L	Setze Bit 2 im L-Register
CB DE	SET	3,(HL)	Setze Bit 3 in (HL)
DD CB di DE	SET	3,(IX+di)	Setze Bit 3 in (IX+di)
FD CB di DE	SET	3,(IY+di)	Setze Bit 3 in (IY+di)
CB DF	SET	3,A	Setze Bit 3 im Akku
CB D8	SET	3,B	Setze Bit 3 im B-Register
CB D9	SET	3,C	Setze Bit 3 im C-Register
CB DA	SET	3,D	Setze Bit 3 im D-Register
CB DB	SET	3,E	Setze Bit 3 im E-Register
CB DC	SET	3,H	Setze Bit 3 im H-Register
CB DD	SET	3,L	Setze Bit 3 im L-Register
CB E6	SET	4,(HL)	Setze Bit 4 in (HL)
DD CB di E6	SET	4,(IX+di)	Setze Bit 4 in (IX+di)
FD CB di E6	SET	4,(IY+di)	Setze Bit 4 in (IY+di)
CB E7	SET	4,A	Setze Bit 4 im Akku
CB E0	SET	4,B	Setze Bit 4 im B-Register
CB E1	SET	4,C	Setze Bit 4 im C-Register
CB E2	SET	4,D	Setze Bit 4 im D-Register
CB E3	SET	4,E	Setze Bit 4 im E-Register
CB E4	SET	4,H	Setze Bit 4 im H-Register
CB E5	SET	4,L	Setze Bit 4 im L-Register
CB EE	SET	5,(HL)	Setze Bit 5 in (HL)
DD CB di EE	SET	5,(IX+di)	Setze Bit 5 in (IX+di)
FD CB di EE	SET	5,(IY+di)	Setze Bit 5 in (IY+di)
CB EF	SET	5,A	Setze Bit 5 im Akku
CB E8	SET	5,B	Setze Bit 5 im B-Register
CB E9	SET	5,C	Setze Bit 5 im C-Register
CB EA	SET	5,D	Setze Bit 5 im D-Register
CB EB	SET	5,E	Setze Bit 5 im E-Register
CB EC	SET	5,H	Setze Bit 5 im H-Register
CB ED	SET	5,L	Setze Bit 5 im L-Register
CB F6	SET	6,(HL)	Setze Bit 6 in (HL)
DD CB di F6	SET	6,(IX+di)	Setze Bit 6 in (IX+di)
FD CB di F6	SET	6,(IY+di)	Setze Bit 6 in (IY+di)
CB F7	SET	6,A	Setze Bit 6 im Akku
CB F0	SET	6,B	Setze Bit 6 im B-Register
CB F1	SET	6,C	Setze Bit 6 im C-Register
CB F2	SET	6,D	Setze Bit 6 im D-Register
CB F3	SET	6,E	Setze Bit 6 im E-Register
CB F4	SET	6,H	Setze Bit 6 im H-Register
CB F5	SET	6,L	Setze Bit 6 im L-Register
CB FE	SET	7,(HL)	Setze Bit 7 in (HL)
DD CB di FE	SET	7,(IX+di)	Setze Bit 7 in (IX+di)
FD CB di FE	SET	7,(IY+di)	Setze Bit 7 in (IY+di)
CB FF	SET	7,A	Setze Bit 7 im Akku
CB F8	SET	7,B	Setze Bit 7 im B-Register
CB F9	SET	7,C	Setze Bit 7 im C-Register
CB FA	SET	7,D	Setze Bit 7 im D-Register
CB FB	SET	7,E	Setze Bit 7 im E-Register
CB FC	SET	7,H	Setze Bit 7 im H-Register
CB FD	SET	7,L	Setze Bit 7 im L-Register
CB 26	SLA	(HL)	Arithmetischer Links-Shift von (HL)
DD CB di 26	SLA	(IX+di)	Arithmetischer Links-Shift von (IX+di)
FD CB di 26	SLA	(IY+di)	Arithmetischer Links-Shift von (Iy+di)
CB 27	SLA	A	Arithmetischer Links-Shift des Akkus
CB 20	SLA	B	Arithmetischer Links-Shift des B-Registers
CB 21	SLA	C	Arithmetischer Links-Shift des C-Registers
CB 22	SLA	D	Arithmetischer Links-Shift des D-Registers
CB 23	SLA	E	Arithmetischer Links-Shift des E-Registers
CB 24	SLA	H	Arithmetischer Links-Shift des H-Registers
CB 25	SLA	L	Arithmetischer Links-Shift des L-Registers
CB 2E	SRA	(HL)	Arithmetischer Rechts-Shift von (HL)
DD CB di 2E	SRA	(IX+di)	Arithmetischer Rechts-Shift von (IX+di)
FD CB di 2E	SRA	(IY+di)	Arithmetischer Rechts-Shift von (IY+di)
CB 2F	SRA	A	Arithmetischer Rechts-Shift des Akkus
CB 28	SRA	B	Arithmetischer Rechts-Shift des B-Registers
CB 29	SRA	C	Arithmetischer Rechts-Shift des C-Registers
CB 2A	SRA	D	Arithmetischer Rechts-Shift des D-Registers
CB 2B	SRA	E	Arithmetischer Rechts-Shift des E-Registers
CB 2C	SRA	H	Arithmetischer Rechts-Shift des H-Registers
CB 2D	SRA	L	Arithmetischer Rechts-Shift des L-Registers
CB 3E	SRL	(HL)	Logischer Rechts-Shift von (HL)
DD CB di 3E	SRL	(IX+di)	Logischer Rechts-Shift von (IX+di)
FD CB di 3E	SRL	(IY+di)	Logischer Rechts-Shift von (IY+di)
CB 3F	SRL	A	Logischer Rechts-Shift des Akkus
CB 38	SRL	B	Logischer Rechts-Shift des B-Registers
CB 39	SRL	C	Logischer Rechts-Shift des C-Registers
CB 3A	SRL	D	Logischer Rechts-Shift des D-Registers
CB 3B	SRL	E	Logischer Rechts-Shift des E-Registers
CB 3C	SRL	H	Logischer Rechts-Shift des H-Registers
CB 3D	SRL	L	Logischer Rechts-Shift des L-Registers
DD 96 di	SUB	(IX+di)	Subtraktion von (IX+di)
FD 96 di	SUB	(IY+di)	Subtraktion von (IY+di)
DD AE di	XOR	(IX+di)	Logisches Exklusiv-Oder mit (IX+di)
FD AE di	XOR	(IY+di)	Logisches Exklusiv-Oder mit (IY+di)

Firmware-Aufruf unter CP/M

Die Einsprungadressen der Schneider-Firmware-Routinen (zum Beispiel zur Grafikausgabe) im Bereich ab der Adresse BB00 hex stehen auch unter CP/M 2.2 zur Verfügung – und das in gleicher Weise auf allen drei Schneider-Computern.

Beim Aufruf müssen folgende Bedingungen erfüllt sein:
1. Der »Stackpointer« zeigt auf das »zentrale RAM« im Bereich von 4000 bis BFFF hex.
2. Im F'-Register ist das Carry-Flag zurückgesetzt.
3. Im BC'-Register steht ein Wert, der einige Systemeinstellungen bestimmt, die ROM/RAM-Auswahl, Bildschirmmodus und Interruptunterdrückung. Im Normalfall ist das bei CP/M im Modus 2 der Wert 7F86 hex, im Modus 1 der Wert 7F85 hex und im Modus 0 der Wert 7F84 hex. Bei bestimmten Speichererweiterungstypen können sich diese Werte auch ändern.

Im Basic-Betriebssystem sind diese Bedingungen automatisch erfüllt. Entgegen der landläufigen Meinung dürfen unter CP/M aber alle Register verändert werden. Viele CP/M-Programme nützen das auch aus. Nur wenn das BIOS von sich aus Firmware-Routinen aufruft (zum Beispiel beim Interrupt), speichert es automatisch die alten Werte im BC'- und F'-Register und rekonstruiert die Soll-Werte. Das ist der Grund, warum CP/M-Programme, die Firmware-Routinen verwenden, oft nicht korrekt arbeiten.

Der Fast/ Slow-Modus

Wenn Sie schon mit »SETUP.COM« experimentiert haben, wissen Sie, daß Sie dort zwischen »FAST«- und »SLOW«-Modus wählen können. Die oben erwähnten Fakten gelten aber nur für den »SLOW«-Modus. Wenn sich Ihr CPC im »FAST«-Modus befindet, werden die Register vor BIOS-internen Firmware-Routinen nicht mehr restauriert und es steht – wie unter Basic – nur ein eingeschränkter Registersatz zur freien Verfügung. Die Register F' und BC' dürfen dann überhaupt nicht mehr verändert werden. Bei Programmen, die ursprünglich für 8080-Computer entwickelt wurden, bedeutet das keine



Viele Schneider-Besitzer freuen sich über die nützlichen Firmware-Routinen ihres Computers. Daß und wie man diese aber unter CP/M, und sogar ohne Assemblerkenntnisse unter Turbo-Pascal benutzen kann, wissen nur wenige.

Einschränkung, da der 8080 den zweiten Registersatz nicht kennt. Bei Verwendung einer Vortex-Diskettenstation können Sie »FAST« übrigens nicht wählen: Der Computer befindet sich dort immer im »SLOW«-Modus.

Die »FAST«/»SLOW«-Umschaltung können Sie auch außerhalb des SETUP-Programms durchführen. Unter CP/M befindet sich an der Adresse BE80 hex eine weitere Sprungtabelle. Durch »ld a,FF hex:call BE9E hex« wird der »FAST«-Modus eingeschaltet und durch »xor a:call BE9E hex« der »SLOW«-Modus.

Die richtige Belegung der BC'- und F'-Register ist eine zusätzliche Fehlerquelle und bringt Programme oft aus scheinbar unerfindlichen Gründen zum Absturz. Deshalb gibt es im Betriebssystem eine kleine Hilfe. Durch »call BE9B hex« wird die Firmware-Routine, deren Adresse sich in den zwei dem Aufruf folgenden Speicherzellen befindet, ausgeführt. Die korrekte Versorgung des alternativen Registersatzes wird dabei automatisch durchgeführt (natürlich nur im »SLOW«-Modus). Das Programm zum Setzen eines Punktes sieht folglich so aus:

```
ld    de,x-Koordinate
ld    hl,y-Koordinate
call  BE9B h
dw    BBEA h
```

Die Sprungtabelle im Bereich von BE80 bis BEA6 hex ermöglicht den Aufruf von zusätzlichen BIOS-Routinen, die nur unter dem CP/M der Schneider-Computer existieren. Mit diesen Einsprüngen können, unabhängig von der Version des Disketten-ROMs, zusätzliche Systemsteuerungen durchgeführt werden. Unverständlicherweise wird von diesen Routinen der alternative Registersatz nicht restauriert. Einfache Routinen, die nur Änderungen im RAM vornehmen, funktionieren trotzdem. Probleme gibt es allerdings bei den Routinen, die sich ihrerseits auf die gewöhnlichen Firmware-Routinen stützen, zum Beispiel während der Wartezeit beim Starten des Diskettenmotors. Hierbei müssen die Register BC' und F' weiterhin manuell versorgt werden. Dabei dürfen Sie aber den Stackpointer nicht vergessen.

Mit Hilfe der Anweisung »Inline« können in Turbo-Pascal-Programme direkt Maschinencode-Routinen eingefügt werden. In Klammern stehen dann, getrennt durch Schrägstriche, die einzelnen Bytes des Maschinenprogramms. Also etwa

```
inline (.../$cd/$5a/$bb/...)
```

Turbo-Pascal stark verbessert

Die einzelnen Bytes erlauben eine sowohl hexadezimale als auch dezimale Angabe. Ist der Wert zwischen zwei Schrägstrichen größer als 255, wird er vom Compiler automatisch in zwei Maschinensprache-Byte übersetzt. Diese Regel hat nur zwei Ausnahmen, die aber eine große Arbeitserleichterung darstellen.
1. Ein Variablenname in einer Inline-Anweisung ergibt im Maschinencode-Programm unabhängig vom Wert und vom Typ (es sind nicht nur Zahlen erlaubt) immer genau zwei Byte. Diese enthalten nicht den Wert der Variablen, sondern die Adresse, an der die Variable im Speicher steht. Wenn Sie den Wert der Integer-Variablen »<Zahl>« im Maschinencode-Programm verarbeiten wollen, dürfen Sie also nicht

```
inline(.../$21/⟨Zahl⟩/...)
```

für »ld hl,<Zahl>« schreiben, sondern Sie müssen

```
inline(.../$2a/⟨zahl⟩/...
```

für »ld hl,(<Zahl>)« benutzen. Diese Methode hat den Vorteil, daß nicht nur Integervariablen, sondern beliebige Typen verwendet werden können und die Maschinencode-Routine auch neue Werte in eine Variable schreiben und ans Hauptprogramm zurückgeben kann.
2. Während der Entwicklung des Maschinencode-Programms ist noch nicht abzusehen, in welchem Speicherbereich es einmal endgültig stehen wird. Absolute Sprünge sind so nicht

empfehlenswert. Deshalb kann in einer Inline-Anweisung auch ein Sternchen (»*«) eingesetzt werden. Dieses repräsentiert zwei Byte, die den aktuellen Wert des Programmzählers beinhalten. Zusätzlich darf zum Sternchen noch eine Konstante addiert oder abgezogen werden. Die Assemblerzeile zum Überspringen der nächsten 17 (nicht 20) Byte mit einem absoluten bedingten Sprung, sieht zum Beispiel so aus:

```
marke: jp nz,marke+20
```

In einer Inline-Anweisung müssen Sie statt dessen schreiben:

```
inline(.../$c2/*+20/...)
```

Diese Regeln genügen bereits, um beliebige Maschinencode-Programme in ein Turbo-Pascal-Programm zu integrieren.

Mit Hilfe von »Inline« lassen sich auch aus einem Pascal-Programm ohne Maschinensprachkenntnisse Firmware-Routinen aufrufen. Das gilt nicht nur für die neue Version 3.0, sondern auch für das alte Turbo-Pascal 2.0. Eine Lösung dazu zeigt Listing 1.

```
  (* CPC-Firmwareroutinen aufrufen; Version fuer keine Speichererweiterung *)

type register = record
                  f:byte;
                  a:byte;
                  bc:integer;
                  de:integer;
                  hl:integer
                end;

procedure firmware(var regvar: register; adress: integer);
  var spvar: integer;
  begin
    inline($2a/adress/$22/*+19/$ed/$73/spvar/$f3/$ed/$7b/regvar/$f1/$c1/$d1/
      $e1/$fb/$cd/$9b/$be/$00/$00/$f3/$e5/$d5/$c5/$f5/$ed/$7b/spvar/$fb)
  end;

procedure exbios(var regvar: register; adress: integer);
  var spvar: integer;
  begin
    inline($2a/adress/$22/*+19/$ed/$73/spvar/$f3/$ed/$7b/regvar/$f1/$c1/$d1/
      $e1/$fb/$cd/$9b/$be/$00/$00/$f3/$e5/$d5/$c5/$f5/$ed/$7b/spvar/$fb)
  end;
```

Listing 1. Binden Sie Maschinencode-Routinen in Ihr Turbo-Pascal-Programm ein

Das erweiterte Schneider-BIOS

Vor dem Aufruf einer Firmware-Routine müssen die Übergabeparameter in die CPU-Register übertragen und die Ergebnisse nach der Bearbeitung an das Pascal-Programm zurückgegeben werden. Deshalb wird mit »type« der neue Variablentyp »register« definiert, in dem für jedes CPU-Register ein Speicherplatz reserviert ist. Unmittelbar vor dem Aufruf einer Firmware-Routine wird der Inhalt der übergebenen Registervariablen in die Register des Z80 übertragen. Nach der Ausführung enthält die Registervariable die neuen CPU-Registerinhalte, so daß sie vom Pascal-Programm leicht abgefragt werden können.

Auf diese Weise können Sie Firmware-Routinen auch verwenden, wenn Sie in Maschinensprache völlig unbedarft sind. In manchen Veröffentlichungen (zum Beispiel im 2. Happy-Computer-Schneider-Sonderheft oder im Schneider-Firmware-Handbuch) finden Sie Informationen zu diesen Routinen oft in der folgenden Form:
Zeichenausgabe am Bildschirm:
Adresse: BB5A hex
Übergabeparameter: Register a: Zeichencode
Ergebnis: –

Auch wenn Sie bisher nicht wußten, was diese Angaben im Detail bedeuten, können Sie in Pascal jetzt einfach schreiben:

```
regvar.a:=ord('⟨Zeichen⟩');firmware(regvar,$bb5a);
```

Nur zwei Dinge sind dabei zu beachten:
1. Die Routineadressen können hexadezimal und dezimal vorgegeben sein. Hinter hexadezimalen Zahlen steht oft ein »h« oder »hex«, um sie von dezimalen Zahlen zu unterscheiden. Gelegentlich sieht man statt dessen auch ein vorangestelltes »&« oder »#«. In Pascal ist zur Kennzeichnung nur ein führendes Dollarzeichen erlaubt (zum Beispiel »$BB5A«). Diese Umsetzung können Sie auch durchführen, wenn Sie ansonsten nicht mit hexadezimalen Zahlen rechnen können – also keine Bange.
2. Die Register B und C können mit der Register-Variablen nur paarweise als BC angesprochen werden. Die Umrechnung gelingt aber ganz einfach durch die Formel:

```
⟨bc-Wert⟩ := ⟨b-Wert⟩ SHL 8 OR ⟨c-Wert⟩.
```

Genau dasselbe gilt für die Register D, E beziehungsweise DE und die Register H, L beziehungsweise HL. Wenn Sie sich die Formel nicht im Kopf merken wollen, können Sie eine einfache Pascal-Funktion

```
functiondoppel(links,rechts:byte):integer:
```

schreiben. Steht in der Angabe zur Firmware-Routine schon von vornherein ein doppelter Registername, erübrigt sich eine Umrechnung.

Mit der neuen Pascal-Prozedur »Firmware« (Listing 1) können Firmware-Routinen aus dem Bereich BB00 bis BD39 hex aufgerufen werden. Zum Aufruf der zusätzlichen BIOS-Routinen im Bereich BE80 bis BEA6 hex, die es nur auf den Schneider-Computern gibt, steht die Prozedur »Exbios« (Listing 1) zur Verfügung. Diese Prozedur kann auf genau dieselbe Weise verwendet werden wie die Prozedur »Firmware«: Übergeben müssen Sie ebenfalls eine Registervariable und die Adresse der BIOS-Routine.

Nicht erlaubt ist dagegen die Routine an der Adresse BE9B hex. Eine sinnvolle Anwendung ist nur mit besonderen Übergabeparametern möglich. Diese Routine wird aber gar nicht benötigt, denn die Prozedur »Firmware« macht genau genommen nichts anderes, als die Arbeitsumgebung für die Routine an der Adresse BE9B hex herzustellen. Das war auch der einzige Grund für ihre Implementierung.

Ein Anwendungsbeispiel der EXBIOS-Routine zeigt das Listing 2 – die Datei »FIRMTEST.PAS«. Es beinhaltet das Grundgerüst – aber schon ablauffähig – eines in Pascal geschriebenen Diskettenmonitors. In genau derselben Weise kann auch die Prozedur »Firmware« verwendet werden, aber natürlich zu anderen Zwecken.

Durch die Zeile

```
{$i Firm44.inc}
```

werden die beiden neuen Pascal-Prozeduren ins Programm integriert, ohne sie dabei noch einmal extra abtippen zu müssen. Sie können sie dann genauso verwenden wie normale Befehle, zum Beispiel »writeln«.

Übergabe von Tabellen an Firmware-Routinen

Die BIOS-Routinen zur Diskettensteuerung reagieren sehr allergisch auf kleinste Tippfehler. Sie wären nicht der erste, der nach derartigen Experimenten ausgerechnet seine Lieblingsdiskette hingemeuchelt hätte. Nur durch Neuformatieren ist eine Diskette in diesen Fällen noch zu retten. Also merken Sie sich: Solange ein Programm mit direkter Diskettensteuerung nicht mit absoluter Sicherheit fehlerfrei läuft, darf es nur auf Disketten getestet werden, die keine wichtigen Dateien enthalten. Vor dem Test sollten Sie die aktuelle Version des Programms natürlich auf einer anderen Diskette speichern.

Manche Firmware-Routinen erwarten im HL-Register die Anfangsadresse

```
program extended_biostest;

{$i firm44.inc}

var regvar:                        register;
    peek, spur, sektor, drive:     byte;
    puffer:                        array[0..511] of byte;
    again:                         char;
    i:                             integer;

begin
  repeat
    write('Laufwerk, Spur und Sektor eingeben: ');
    readln(drive, spur, sektor);
    regvar.bc := sektor;
    regvar.de := spur shl 8 + drive;
    regvar.hl := addr(puffer);
    exbios(regvar, $be89);
    if (regvar.f and 1) = 0
      then
        writeln('Sektor konnte nicht gelesen werden!')
      else
        begin




          for i := 0 to 511 do
            begin
              peek := puffer[i] and 127;
              if (peek<32) or (peek=127)
                then
                  write('.')
                else
                  write(chr(peek))
            end;
          writeln
        end;
    repeat
      write('Nochmal? (j/n) ');
      readln(again)
    until (again='j') or (again = 'n')
  until again<>'j'
end.
```

Listing 2. Ein Disketten-Monitor unter Turbo-Pascal

```
******* Version f}r CPC ohne Speichererweiterung ****************
(Adressen im Bereich von '7Cxx' wurden vom Compiler
 automatisch eingesetzt und k!nnen auch andere Werte haben)

****** Procedur 'Firmware'
1FF3  LD    HL,(7C78)       Adresse der Firmwareroutine
1FF6  LD    (200A),HL       Ins Progamm kopieren
1FF9  LD    (7C72),SP       Stapel retten
1FFD  DI
1FFE  LD    SP,(7C7A)       Adresse der Register-variablen
2002  POP   AF              Record in Register kopieren
2003  POP   BC
2004  POP   DE




2005  POP   HL
2006  EI
2007  CALL  BE9B            Firmwareroutine ausf}hren
200A  DW    0000            Hier steht sp{ter die Routinenadresse
200C  DI
200D  PUSH  HL              Ergebnis an Variable zur}ckgeben
200E  PUSH  DE
200F  PUSH  BC
2010  PUSH  AF
2011  LD    SP,(7C72)       Stackpointer restaurieren
2015  EI

****** Procedure 'GRA MOVE ABSOLUTE'
206C  LD    DE,(7C66)       x- und y-Koordinate
2070  LD    HL,(7C64)
2073  CALL  BE9B            Firmwareroutine ausf}hren
2076  DW    BBC0            Adresse der Routine
```

Listing 3. »Firmware« im Detail

einer Tabelle, die weitere zu verarbeitende Daten enthält. Im Pascal-Programm stehen derartige Daten in der Regel in einem Feld oder einer Zeichenkette. In Turbo-Pascal erhalten Sie die erste Adresse einer Variablen mit der Funktion »addr(<Variablenname>)«. Beachten Sie aber, daß das erste Byte einer Zeichenkettenvariablen nur die Länge der Variablen enthält. Den eigentlichen Text findet man erst ab dem zweiten Byte. Zum Beispiel können Sie die Funktionstaste 128 mit dem Text »Taste128« belegen:

```
textvar := 'Taste128';
regvar.bc := 128 SHL 8 OR mem
[addr(textvar)];
regvar.hl – := addr(textvar) + 1;
firmware(regvar, $BBOF);
```

Mit dem Pendant des Basic-Befehls KEY DEF können Sie übrigens zusätzlich die Funktionstaste 128 statt auf den Zehnerblock auf eine beliebige andere Taste legen.

»mem[<Adresse>]« entspricht PEEK unter Basic und holt in unserem Fall die Länge der Zeichenkette direkt aus einer Speicherzelle. Analog entspricht »mem[<Adresse>]:=<Wert>« einem Basic-POKE.

Wenn Sie mit einer Speichererweiterung arbeiten, muß sich die zu übergebende Tabelle im Speicherbereich 0000 bis 7FFF hex befinden. Das erreichen Sie, indem Sie die Compiler-Option »COM-File« wählen. Dann können Sie mit der Option »Startadresse« eine neue Startadresse des Programms festlegen. Der Bereich zwischen alter und neuer Adresse beginnt bei 1FC9 hex, und damit in der unteren Speicherhälfte. Er eignet sich für beliebige Zwecke. Durch die Zuweisung

```
var ⟨Name⟩ : ⟨Typ⟩ ABSOLUTE
⟨Adresse⟩:
```

können Sie die Variable <Name> in genau diesen Bereich »zwingen«. Ihre Adresse wird also nicht mehr vom Compiler automatisch an das obere Ende des freien Speichers gelegt.

Listing 3 zeigt, wie die Prozedur »Firmware« im Detail funktinoniert. Zunächst wird der Stackpointer in der Variablen »spvar'« zwischengespeichert und danach auf die übergebene Registervariable gerichtet. Die obersten Elemente des neuen Stapels entsprechen dem Inhalt der Registervariablen, so daß sie durch POP-Befehle in die CPU-Register übertragen werden können. Während dessen wird der Interrupt abgeschaltet, damit dieser keine PUSH-Befehle ausführen und damit über das Ende der Registervariablen hinausschreiben kann.

»call BE9B hex« bringt eine beliebige Firmware-Routine zur Ausführung. Die Adresse Routine wurde schon vorher in die reservierten Bytes »dw 0000 hex« übertragen. Bevor (automatisch) ein lokaler Stapel angelegt wird, benötigen der Interrupt und der »call BE9B hex« zusammen niemals mehr als vier Stapelplätze. Aus diesem Grund können die Interrupts wieder freigegeben werden.

Nach der Ausführung werden die neuen CPU-Registerwerte nach derselben Methode wieder in die Registervariable zurück»gePUSHed« und der alte Stackpointerwert restauriert.

Wie schon beschrieben, entfällt beim Aufruf der erweiterten BIOS-Routinen eine Restaurierung des BC'- und des F'-Registers. Deshalb greift man zu dem Trick, alle anderen BIOS-Routinen ebenfalls mit Hilfe der BIOS-Routine an der Adresse BE9B hex aufzurufen, die das dann mit übernimmt. Diese Verwendung wurde von den Entwicklern zwar nicht vorgesehen, aber in diesem Fall funktioniert es einwandfrei. Dieser Trick hat zur Folge, daß sich die Routinen »Firmware« und »Exbios« vollständig gleichen. Bei der Version für eine Speichererweiterung ist das aber nicht mehr der Fall. Deshalb wurden aus Kompatibilitätsgründen trotzdem zwei getrennte Prozeduren beibehalten. »Bitspar-Fanatiker« können aber eine weglassen.

Grafikerweiterung für Turbo-Pascal

Die bisher besprochene Prozedur »Firmware« ist zwar so allgemein, daß wirklich jede Firmware-Prozedur aufgerufen werden kann, bei mehrmaliger Verwendung immer der gleichen Firmware-Routine ist das Verfahren aber etwas umständlich.

Die Datei »GRAPH44.INC« (Listing 4) enthält deshalb spezielle Prozeduren, um die Grafik-Befehle des CPC-Betriebssystems anzusprechen. Da an diese Routinen nicht jedesmal alle acht CPU-Register übergeben werden müssen und die Rückgabe von Registern generell entfällt, sind diese wesentlich kürzer. Dafür lassen sich aber andere Firmware-Routinen nicht damit aufrufen.

Die Grafik-Befehle stehen in einem Pascal-Programm dann zur Verfügung, wenn es am Anfang die Zeile

`{$i GRAPH44.INC}`

enthält. Das nochmalige Abtippen oder speicherplatzfressende Einfügen mit CTRC-KR entfällt dann. Die Befehle können im wesentlichen genauso angewendet werden, wie die entsprechenden ähnlich lautenden Basic-Befehle.

Die Prozedur »Grainit« setzt das Grafiksystem des Computers zurück: Windowgrenzen werden gelöscht, Farbstiftnummern und Schreibmodi auf den Anfangswert gesetzt. Vergißt man diesen Befehl, ist es aber nicht weiter schlimm (schlimmstenfalls sehen die Linien etwas »komisch« aus). Durch »Grapen« und »Grapaper« wird festgelegt, welcher Farbstift das Zeichnen und Löschen von Grafik-Elementen ausführt. Die Textfarbstifte bleiben davon jedoch unberührt. Durch »Graclear« wird nur der Bildschirmbereich gelöscht, der durch »Grawindow« vorgegeben wurde. Linien (und sonstige Grafikausgaben) werden nur bis zu den Grenzen des durch »Grawindow« vorgegebenen Bereichs gezeichnet, selbst wenn die Koordinaten größere Werte annehmen. »Graorigin« fixiert die Lage des Koordinatenursprungs.

```
(* Graphik-Befehlserweiterung ;Version fuer keine Speichererweiterung *)

procedure grainit;  (* Graphik-System (Modi, Farben usw) zuruecksetzen *)
  begin
    inline($cd/$9b/$be/$ba/$bb)
  end;

procedure gramove (x, y: integer);  (* Graphikcursor absolut setzen *)
  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$c0/$bb)
  end;

procedure gramover (x, y: integer);  (* Graphikcursor relativ setzen *)
  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$c3/$bb)
  end;

procedure graorigin (x, y: integer);  (* Koordinatenursprung waehlen *)
  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$c9/$bb)
  end;

procedure grawindow (x, y, o, u: integer);  (* Graphikfenster waehlen *)





  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$cf/$bb/$ed/$5b/o/$2a/u/$cd/$9b/$be/
      $d2/$bb)
  end;

procedure graclear;  (* Graphikfenster loeschen *)
  begin
    inline($cd/$9b/$be/$db/$bb)
  end;

procedure grapen (c: byte);  (* Zeichenstiftnummer waehlen *)
  begin
    inline($3a/c/$cd/$9b/$be/$de/$bb)
  end;

procedure grapaper (c: byte);  (* Hintergrundstiftnummer waehlen *)
  begin
    inline($3a/c/$cd/$9b/$be/$e4/$bb)
  end;

procedure graplot (x, y: integer);  (* Bildschirmpunkt absolut setzen *)
  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$ea/$bb)
  end;

procedure graplotr (x, y: integer);  (* Bildschirmpunkt relativ setzen *)
  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$ed/$bb)
  end;

procedure graline (x, y: integer);  (* Linie absolut ziehen *)
  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$f6/$bb)
  end;

procedure graliner (x, y: integer);  (* Linie relativ ziehen *)
  begin
    inline($ed/$5b/x/$2a/y/$cd/$9b/$be/$f9/$bb)
  end;
```

Listing 4. Grafik-Befehlserweiterung für Turbo-Pascal

```
(* CPC-Firmwareroutinen aufrufen: Version fuer Vortex-Speichererweiterung *)

type register = record
                  f:byte;
                  a:byte;
                  bc:integer;
                  de:integer;
                  hl:integer
                end;

procedure firmware(var regvar: register; adress: integer);
  var spvar: integer;
  begin
    inline($ed/$73/spvar/$f3/$ed/$7b/regvar/$f1/$c1/$d1/$e1/$fb/$d9/$21/*+$0e/
      $e5/$2a/adress/$23/$23/$23/$e5/$d9/$c3/$42/$f5/$f3/$e5/$d5/$c5/$f5/$ed/
      $7b/spvar/$fb)
  end;

procedure exbios(var regvar: register; adress: integer);
  var spvar: integer;
  begin
    inline($ed/$73/spvar/$f3/$ed/$7b/regvar/$f1/$c1/$d1/$e1/$ed/$7b/spvar/$fb/
      $d9/$21/*+$0d/$e5/$2a/adress/$11/$b9/$35/$19/$e5/$d9/$c9/$f3/$08/$d9/
      $2a/regvar/$11/$08/$00/$19/$f9/$d9/$08/$e5/$d5/$c5/$f5/$ed/$7b/spvar/
      $fb)
  end;
```

Listing 5. Firmware-Routinen und Speichererweiterung: kein Gegensatz

```
********** Version f}r Vortex-Speichererweiterung ***************
(Die Werte im Bereich von 'C3xx' wurden vom Compiler automatisch
berechnet und k|nnen auch andere Werte haben)

********* Prozedur 'Firmware'
1FF3 LD    (C372),SP     Stapel retten
1FF7 DI
1FF8 LD    SP,(C37A)     Adresse des Register-Records
1FFC POP   AF            Register mit Variablemwerten laden
1FFD POP   BC
1FFE POP   DE
1FFF POP   HL
2000 EI
2001 EXX                 Register retten
2002 LD    HL,2011       Fortsetzungsadresse nach Ausf}hrung
2005 PUSH  HL
2006 LD    HL,(C378)     Adresse der Routine
2009 INC   HL            um 3 erh|hen
200A INC   HL
200B INC   HL
200C PUSH  HL            auf Stapel speichern
200D EXX
200E JP    F542          Routine ausf}hren
2011 DI                  Hier Fortsetzung nach Ausf}hrung
2012 PUSH  HL            Register in Variable abspeichern
2013 PUSH  DE
2014 PUSH  BC
2015 PUSH  AF
2016 LD    SP,(C372)     Stapel restaurieren
201A EI

********** Erweitertes BIOS aufrufen
2028 LD    (C368),SP     Stapel abspeichern
202C DI
202D LD    SP,(C370)     Anfangsadrsse des Register-Records
2031 POP   AF            Variablenwerte in Register }bertragen
2032 POP   BC
2033 POP   DE
2034 POP   HL
2035 LD    SP,(C368)     alten Stapel restaurieren
2039 EI
203A EXX
203B LD    HL,2049       Fortsetzungsadresse nach Ausf}hrung
203E PUSH  HL
203F LD    HL,(C36E)     Adresse der Routine
2042 LD    DE,35B9       Sprungtabelle beginn jetzt bei F439
2045 ADD   HL,DE         Versatz korrigieren
2046 PUSH  HL            auf Stapel
2047 EXX
2048 RET                 Routine ausf}hren
2049 DI                  Hier Fortsetzung nach Ausf}hrung
204A EX    AF,AF'        Register retten
204B EXX
204C LD    HL,(C370)     Adresse des Register-REcords
204F LD    DE,0008       Ende des Records
2052 ADD   HL,DE
2053 LD    SP,HL
2054 EXX                 Register restaurieren
2055 EX    AF,AF'
2056 PUSH  HL            In Ergebnisvariable }bertragen
2057 PUSH  DE
2058 PUSH  BC
2059 PUSH  AF
205A LD    SP,(C368)     Stapel restaurieren
205E EI

******* Aufruf von 'GRA MOVE ABSOLUTE'
206C LD    HL,207E       Fortsetzung nach Ausf}hrung
206F PUSH  HL
2070 LD    HL,BBC3       Routinenadresse plus 3
2073 PUSH  HL
2074 LD    DE,(C366)     x- und y-Koordinae
2078 LD    HL,(C364)
207B JP    F542          Routine aufrufen
```

Listing 6. Das Assembler-Listing der Prozedur »Firmware«

Zur Auswahl der Farben und des Bildschirmmodus wurden keine Befehle eingefügt. Diese Aufgaben (und noch andere) werden mit den üblichen »Write«-Anweisungen ausgeführt. Ein

`write(#4,chr(1));`

schaltet beispielsweise den Bildschirm in den Modus 1 um. Durch die Zeile

`write(#28,chr(3),chr(20),chr(26));`

wird der Farbstift mit der Nummer 3 mit einer Farbe gefüllt, die mit den Werten 20 und 26 blinkt. Eine genaue Beschreibung aller Kontrollzeichen steht in jedem CPC-Handbuch unter den Stichworten Steuer-Codes und Kontrollzeichen.

Wenn Sie Assembler beherrschen, sehen Sie die Funktionsweise der Grafik-Routinen ebenfalls im Listing 3, am Beispiel der Prozedur »gramove«. Die anderen funktionieren in der gleichen Weise. Die Übergabeparameter werden jetzt nicht mehr auf dem Umweg über den Stapel in die CPU-Register übertragen, sondern direkt aus den Speicheradressen. Die Rückgabe der Ergebnisregister unterbleibt. Da die Routinenadresse nicht mehr variabel ist, steht sie direkt im Maschinencode-Programm.

Die folgenden Erklärungen und Programme gelten uneingeschränkt, wenn Sie Ihren CPC mit einer Vortex-Speichererweiterung und einem 62KByte-CP/M 2.2 betreiben.

Bei der Verwendung anderer Speichererweiterungstypen ist es zwar möglich, daß der Firmware-Aufruf in prinzipiell ähnlicher Weise erfolgt, einige kleine Änderungen sind aber sicher nötig. Allerdings erfordern diese Änderungen Maschinensprach-Kenntnisse. Ein Versuch mit den unveränderten Programmen kann aber unter keinen Umständen schaden.

Der Systemvektor ist wichtig

Zumindest bei der Vortex-Speichererweiterung befindet sich im Bereich von BB00 bis BD39 hex und im Bereich von BE80 bis BEA0 hex der sogenannte »Systemvektor«. Wenn ein Programm für das normale 44KByte-CP/M entwickelt wurde, konnte es die CPC-Firmware-Routinen uneingeschränkt verwenden. Läuft dieses Programm jetzt unter dem 62KByte-CP/M, gelingt ein Firmwareaufruf nur über diesen Systemvektor. Er leitet den Firmwareaufruf so um, daß er schließlich auf den »echten« Firmware-Einsprung in der System-Speicherbank trifft, für den er eigentlich gedacht war. Für das Hauptprogramm macht es so (glücklicherweise) keinen Unterschied, ob es unter dem normalen oder dem erweiterten CP/M läuft.

Die originalen Firmware-Routinen lagen hinter dem Ende des freien Arbeitsspeichers, so daß sie vor Überschreiben geschützt waren. Der Systemvektor liegt aber mitten in der TPA. Wenn ein Pascal-Programm diesen Systemvektor beim Betrieb nicht überschreiben soll, muß es mit der Compiler-Option »COM-File« und einer verminderten Endadresse übersetzt werden. Das Testen des Programms ist in diesem Fall nicht einfach mit »R« möglich, sondern nur außerhalb des Pascal-Systems, nach einem Warmstart. Wenn diese Bedingungen erfüllt sind, können allerdings das normale »GRAPH44.INC«- und das »FIRM44.INC«-Pro-

gramm weiterverwendet werden. Über 18 KByte des Speichers werden dabei aber verschwendet.

Es ist nun aber auch möglich, die Firmware-Routinen aufzurufen, ohne den Systemvektor zu benutzen. Dann stehen auch für Pascal-Programme mit Grafikausgabe über 57 KByte freier Arbeitsspeicher zur Verfügung.

Firmware-Aufruf ohne Systemvektor-verwendung

Bei einem 62KByte-CP/M stehen im Bereich ab BB00 hex hintereinander lauter gleiche Befehle »call F542 hex«. Beim Aufruf beliebiger Firmware-Routinen wird die Bearbeitung an immer derselben Adresse fortgesetzt. Allerdings enthält dann das oberste Stapelelement die Rückkehradresse, und damit den um 3 erhöhten Wert der eigentlichen Firmware-Adresse. Durch »POPen« des Stapels kann das Unterprogramm feststellen, welche Routine im System-RAM aufgerufen werden soll. Erst das zweitoberste Stapelelement enthält die gewöhnliche Rückkehradresse ins Hauptprogramm.

Eine Pascal-Prozedur »Firmware« kann unter Umgehung des Systemvektors die Adresse F542 hex direkt aufrufen, wenn sie diesen Stapelaufbau genau nachbildet. Genau das macht die Prozedur im Listing 5. Der Vorspann und der Nachspann entsprechen genau dem des Firmware-Aufrufs der 44KByte-Version. Das Ganze verdeutlicht das Assemblerlisting der Firmware62KByte-Prozedur im Listing 6.

Auch für die 62KByte-Version lohnt es sich, für den Aufruf der Grafik-Routinen Kurzfassungen der Firmware-Prozedur herzustellen. Die Lösung zeigt Listing 7. Gegenüber der 44KByte-Version sind die Routinen ein bißchen umfangreicher geworden. An der Bedienung hat sich aber nichts geändert.

Ein Pascal-Programm, das ursprünglich für die 44KByte-Betriebssystemversion geschrieben wurde, läßt sich ganz einfach an die 62KByte-Version anpassen, indem man im Vorspann des Quellprogramms das »44« in »{$i FIRM44.INC}« beziehungsweise »[$i GRAPH44.INC]« in »62« umbenennt.

Die Adresse F542 hex und damit die gesamte Umgehung des Systemvektors ist leider von der aktuellen Betriebssystemversion abhängig. Mit Hilfe des Programms »DDT.COM« auf Ihrer CP/M-Systemdiskette können Sie aber feststellen, welche Betriebssystemversion Sie haben. Wenn Sie nach dem Start des DDT-Programms durch die Eingabe von »LBB00« in jeder Ausgabezeile das Ergebnis »CALL F542« erhalten, ist alles in Ordnung. Wenn nicht, müssen Sie die Pascal-Prozedur »Firmware« anpassen.

```
(* Graphik-Befehlserweiterung ;Version fuer Vortex-Speichererweiterung *)

procedure grainit;  (* Graphik-System (Modi, Farben usw) zuruecksetzen *)
  begin
    inline($21/*+$0a/$e5/$21/$BD/$bb/$e5/$c3/$42/$f5)
  end;

procedure gramove (x, y: integer);  (* Graphikcursor absolut setzen *)
  begin
    inline($21/*+$11/$e5/$21/$c3/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5)
  end;

procedure gramover (x, y: integer);  (* Graphikcursor relativ setzen *)
  begin
    inline($21/*+$11/$e5/$21/$c6/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5)
  end;

procedure graorigin (x, y: integer);  (* Koordinatenursprung waehlen *)
  begin
    inline($21/*+$11/$e5/$21/$cc/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5)
  end;

procedure grawindow (x, y, o, u: integer);  (* Graphikfenster waehlen *)
  begin
    inline($21/*+$11/$e5/$21/$d2/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5/
      $21/*+$11/$e5/$21/$d5/$bb/$e5/$ed/$5b/o/$2a/u/$c3/$42/$f5)
  end;

procedure graclear;  (* Graphikfenster loeschen *)
  begin
    inline($21/*+$0a/$e5/$21/$de/$bb/$e5/$c3/$42/$f5)
  end;

procedure grapen (c: byte);  (* Zeichenstiftnummer waehlen *)
  begin
    inline($21/*+$0d/$e5/$21/$e1/$bb/$e5/$3a/c/$c3/$42/$f5)
  end;

procedure grapaper (c: byte);  (* Hintergrundstiftnummer waehlen *)
  begin
    inline($21/*+$0d/$e5/$21/$e7/$bb/$e5/$3a/c/$c3/$42/$f5)
  end;

procedure graplot (x, y: integer);  (* Bildschirmpunkt absolut setzen *)
  begin
    inline($21/*+$11/$e5/$21/$ed/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5)
  end;

procedure graplotr (x, y: integer);  (* Bildschirmpunkt relativ setzen *)
  begin
    inline($21/*+$11/$e5/$21/$F0/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5)
  end;

procedure graline (x, y: integer);  (* Linie absolut ziehen *)
  begin
    inline($21/*+$11/$e5/$21/$f9/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5)
  end;

procedure graliner (x, y: integer);  (* Linie relativ ziehen *)
  begin
    inline($21/*+$11/$e5/$21/$fc/$bb/$e5/$ed/$5b/x/$2a/y/$c3/$42/$f5)
  end;
```

Listing 7. Alle Grafik-Routinen für das »große« CP/M

Anpassung an andere BIOS-Versionen

Dazu nehmen Sie die beim DDT-Test hinter den »CALLs« angezeigte Adresse und vertauschen zunächst die hinteren und die vorderen Ziffern (in dieser Version also zu »42 F5«). Im letzten Drittel der Inline-Anweisung in der Firmware-Routine finden Sie die Zeichenkette »/$c3/$42/$f5«. Die Zahlen »42« und »f5« ersetzen Sie jetzt durch Ihre eigenen Werte. Beachten Sie dabei unbedingt die Vertauschung der Ziffern.

Dieselbe Änderung müssen Sie auch in den 62KByte-Versionen aller Grafik-Routinen vornehmen. Wichtig ist, daß die Zeichenkette »/$c3/$42/$f5« in der Prozedur »Grawindow« zweimal vorkommt, und auch zweimal geändert werden muß.

Von den erweiterten BIOS-Routinen gibt es im 62KByte-CP/M zwei Versionen. Einmal als Originalversionen im Disketten-ROM und einmal als Kopien im neuen BIOS, im Bereich von F400 bis FFFF hex der im CP/M sichtbaren Speicherbank.

Die Original-Routinen können Sie genau wie im 44KByte-CP/M mit Hilfe der Prozedur »Firmware« aufrufen. In der Regel ist das aber nicht sinnvoll. Alle Tabellen, die an die Routinen übergeben werden, müssen sich dann im Speicherbereich zwischen 0000 und 7FFF hex befinden. Wenn Sie aber zum Beispiel Daten abspeichern, die sich in der oberen Speicherhälfte befinden oder auch die RAM-Disc verwenden wollen, müssen Sie zu den Routinen greifen, die speziell an die Speichererweiterung angepaßt sind. Diese sind auch über den »normalen« Systemvektor zu erreichen.



Die Sprungtabelle im Bereich ab BE80 hex ist nur eine Kopie der originalen Sprungtabelle. Das Original befindet sich im Bereich von F439 bis F459 hex. Statt dem Systemvektor kann also durch eine einfache Addition des Versatzes gleich die richtige Routine aufgerufen werden. Das BC- und F-Register sowie ein lokaler Stapel, brauchen bei den neuen BIOS-Routinen nicht restauriert zu werden. Die umständliche Prozedur mit dem »call BE9B hex«-Befehl kann also entfallen. Dafür reicht jetzt aber ein vier Einträge großer Stapelbereich nicht aus. Vor der Ausführung der Routine muß also der Originalstapel wiederhergestellt werden. Das bedingt wiederum die Neuberechnung der Registervariablen-Adresse nach der Ausführung.

Glücklicherweise befindet sich die erweiterte BIOS-Tabelle an einer »ausgezeichneten« Stelle, direkt hinter den gewöhnlichen BIOS-Routinen. Es ist also nicht zu erwarten, daß sich bei einer geänderten Betriebssystemversion andere Einsprungadressen ergeben. Es ist wirklich unverständlich, daß beim Systemvektor der »gewöhnlichen« BIOS-Routinen nicht genauso vorgegangen wurde.

Zwei neue zusätzliche BIOS-Routinen

Für die Routinen, die ursprünglich über die Adressen BEA1 und BEA4 hex zu erreichen waren, ist im Systemvektor kein Ersatz vorhanden. Wenn Sie gerade diese benötigen, müssen Sie über den Umweg der Routine »Firmware« in jedem Fall die ROM-Routinen aufrufen. Dasselbe gilt für das Ein/Ausschalten der schnellen Bildschirmausgabe mit der Routine an der Adresse BE86 hex (bei der Benutzung eines VDOS 2.0-ROMs). Die übrigen Funktionen dieser Routine erfüllt aber auch der »Ersatz« an der Adresse F43F hex.

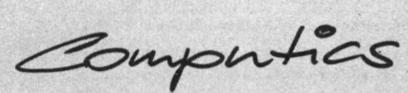
Gemischte Aufrufe der erweiterten BIOS-Routinen, abwechselnd über die Prozedur »Firmware« und über die Prozedur »Exbios«, sollten Sie mit Vorsicht genießen. Systemeinstellungen, die mit einer ROM-Routine durchgeführt wurden, wirken nur bei der Verwendung anderer ROM-Routinen. Umgekehrt wirken Systemeinstellungen mit den neuen RAM-Routinen auch nur auf die Ausführung der anderen RAM-Routinen.

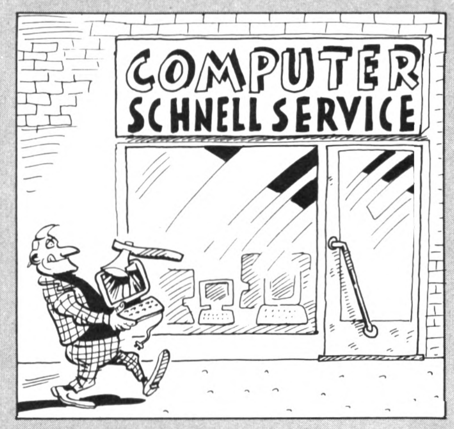
Wenn Sie im Listing 2 die Zeile {$i FIRM44.INC} zu {$i FIRM62.INC} ändern, können Sie die 62KByte-Versionen gleich testen.

An den Adressen F433 und F436 hex befinden sich die Aufrufe von zwei neuen Routinen. Durch »call F433 hex« wird der Druckerspooler ein/ausgeschaltet und durch »call F436 hex« die RAM-Disk formatiert (falls vorhanden).

Aus Pascal aufgerufen werden können die beiden Routinen mit »exbios (regvar,$BE7A);« und »exbios(regvar, $BE7D)«. Die beiden Adressen sind natürlich nur symbolische Adressen: In Wirklichkeit wird der Systemvektor nicht benutzt. Und das war ja auch der Sinn der Sache!

(Helmut Tischer/hg)



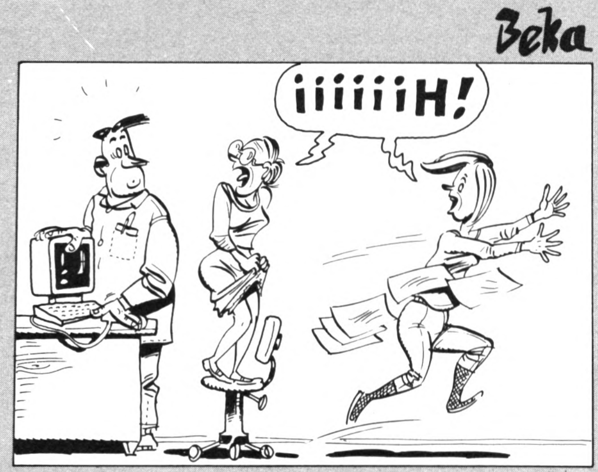












Depot-Händler

Tragen Sie Ihre Buchbestellung auf eine Postkarte ein und schicken diese an einen Depothändler in Ihrer Nähe oder an Ihren Buchhändler.

Buchhandlung Herder, Kurfürstendamm 69
1000 Berlin 15, Tel. (030) 8835002,
BTX *921782#
Computare Fachbuchhandlung, Keithstraße 18
1000 Berlin 30, Tel. (030) 2139021
Thalia Buchhaus, Große Bleichen 19
2000 Hamburg 36, Tel. (040) 3005050
Boysen + Maasch, Hermannstraße 31
2000 Hamburg 1, Tel. (040) 30050515
Electro-Data, Wilhelm-Heidsiek-Straße 1
2190 Cuxhaven, Tel. (04721) 51288
Buchhandlung Muehlau, Holtenauer Straße 116
2300 Kiel, Tel. (0431) 85085
ECL, Norderstraße 94-96
2390 Flensburg, Tel. (0461) 28181
Buchhandlung Weiland, Königstraße 79
2400 Lübeck, Tel. (0451) 74006-09
Buchhandlung Storm, Langenstraße 10
2800 Bremen 1, Tel. (0421) 321523
Buchhandlung Lohse-Eissing, Marktstraße 38
2940 Wilhelmshaven, Tel. (04421) 41687
Buchhandlung Schmorl u. v. Seefeld, Bahnhofstraße 13
3000 Hannover 1, Tel. (0511) 327651
Buchhandlung Graff, Neue Straße 23
3300 Braunschweig, Tel. (0531) 49271
Deuerlich'sche Buchhandlung, Weender Straße 33
3400 Göttingen, Tel. (0551) 56868
Buchhandlung an der Hochschule, Holländische Straße 22
3500 Kassel, Tel. (0561) 83807
Stern Verlag, Friedrichstraße 24-26
4000 Düsseldorf, Tel. (0211) 373033
Buchhandlung Baedeker, Kettwiger Straße 33-35
4300 Essen 1, Tel. (0201) 221381
Regensberg'sche Buchhandlung, Alter Steinweg 1
4400 Münster, Tel. (0251) 40541-5
Buchhandlung Acker, Johannisstraße 51
4500 Osnabrück, Tel. (0541) 28488
Buchhandlung Brockmeyer, Querenburger Höhe 281/Unicenter
4630 Bochum, Tel. (0234) 701360
Buchhandlung Meier + Weber, Warburger Straße 98
4790 Paderborn, Tel. (05251) 63172
Buchhandlung Phönix GmbH, Oberntorwall 25
4800 Bielefeld 1, Tel. (0521) 58306-38
Buchhandlung Gonski, Neumarkt 24
5000 Köln 1, Tel. (0221) 210528
Mayer'sche Buchhandlung, Ursulinerstraße 17-19
5100 Aachen, Tel. (0241) 48142
Buchhandlung Behrendt, Am Hof 5a
5300 Bonn 1, Tel. (0228) 658021
Buchhandlung Cusanus, Schloßstraße 12
5400 Koblenz, Tel. (0261) 36239
Akad. Buchhandlung Interbook, Fleischstraße 61-65
5500 Trier, Tel. (0651) 43596
Buchhandlung W. Finke, Kipdorf 32
5600 Wuppertal 1, Tel. (0202) 454220
Buchhandlung Balogh, Sandstraße 1
5900 Siegen, Tel. (0271) 55298-9
Buchhandlung Naacher, Steinweg 3
6000 Frankfurt 1, Tel. (069) 298050
Buchhandlung Wellnitz, Lautenschlägerstraße 4
6100 Darmstadt, Tel. (06151) 76548
Buchhandlung Feller + Gecks, Friedrichstraße 31
6200 Wiesbaden, Tel. (06121) 304911
Ferber'sche UNI-Buchhandlung, Seltersweg 83
6300 Gießen, Tel. (0641) 12001
Sozialwissenschaftliche Fachbuchhandlung, Friedrichstraße 24
6400 Fulda, Tel. (0661) 75077
Gutenberg Buchhandlung, Große Bleiche 29
6500 Mainz, Tel. (06131) 37011
Buchhandlung Bock + Seip, Futterstraße 2
6600 Saarbrücken, Tel. (0681) 30677
Buchhandlung Wilhelm Hofmann, Bismarckstraße 98
6700 Ludwigshafen, Tel. (0621) 516001
Buchhandlung Loeffler, B 1,5
6800 Mannheim 1, Tel. (0621) 28912
Buchhandlung Stehn, Bahnhofstraße 13
7000 Stuttgart 50, Tel. (0711) 561476
Buchhandlung am Markt, Kramstraße 6
7100 Heilbronn, Tel. (07131) 68682
PCB Micro-Computer, Oskar-Kalbfell-Platz 8
7410 Reutlingen, Tel. (07121) 270443
UNI Buchhandlung Kellner + Moessner, Kaiserstraße 18
7500 Karlsruhe, Tel. (0721) 691436
Buchhandlung Roth, Hauptstraße 45
7600 Offenburg, Tel. (0781) 22097
Rombach Center, Bertholdstraße 10
7800 Freiburg, Tel. (0761) 49091
Fachbuchhandlung Hofmann, Hirschstraße 4
7900 Ulm, Tel. (0731) 60949
Schauties Elektronik, Bachstraße 52
7980 Ravensburg, Tel. (0751) 26138
Buchhandlung Hugendubel, Marienplatz
8000 München 2, Tel. (089) 2389-1
Computerbücher am Obelisk, Barerstraße 32-34
8000 München 2, Tel. (089) 282383
Pele's Computerbücher, Schillerstraße 17
8000 München 2, Tel. (089) 555229
Universitätsbuchhandlung Lachner, Theresienstraße 43
8000 München 2, Tel. (089) 521340
Buchhandlung Schönhuber, Theresienstraße 6
8070 Ingolstadt, Tel. (0841) 33146/47
Computerstudio Gertrud Friedrich, Ludwigstraße 3
8220 Traunstein, Tel. (0861) 14767
Buchhandlung Pustet, Kl. Exerzierplatz 4
8390 Passau, Tel. (0851) 56945
Buchhandlung Pustet, Gesandtenstraße 6
8400 Regensburg, Tel. (0941)53061
Buchhandlung Dr. Büttner, Adlerstraße 10-12
8500 Nürnberg, Tel. (0911) 232318
Computer-Center-Burger, Leimitzer Straße 11-13
8670 Hof, Tel. (09281) 40075
Sortiments- u. Bahnhofsbuchh. J. Strykowski, Bahnhofplatz 4
8700 Würzburg, Tel. (0931) 54389
Buchhandlung Pustet, Grottenau 4
8900 Augsburg, Tel. (0821) 35437
Kemptener Fachsortiment, Salzstraße 30
8960 Kempten, Tel. (0831) 14413

Belgien:
Eicher Micro & Personal Computer, Hünningen 56-58
B-4780 St. Vith, Tel. (080) 227393

Luxemburg:
Librairie Promoculture, 14, rue Duchscher (Pl. de Paris)
L-1011 Luxembourg-Gare, Tel. 480691, Telex 3112

Schweiz:
Buchhandlung Meissner, Bahnhofstraße 41
5000 Aarau, Tel. (064) 247151
Bücher Balmer, Neugasse 12
6300 Zug, Tel. (042) 214141
Buchhandlung Enge, Bleicherweg 56
8002 Zürich, Tel. (01) 2012078
Buchhandlung Orell Füssli, Pelikanstraße 10
8022 Zürich, Tel. (01) 2118011
Freihofer AG, Wissenschaftliche Buchhandlung, Universitätsstr. 11
8033 Zürich, Tel. (01) 3634282
Buchhandlung am Rößlitor, Webergasse 5
9001 St. Gallen, Tel. (071) 228726

Markt&Technik

Unternehmensbereich Buchverlag

Hans-Pinsel-Straße 2, 8013 Haar bei München

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael Scharfenberger (sc)
Leitender Redakteur: Michael Lang (lg)
Redakteure: Andreas Hagedorn (hg), Thomas Jacobi (ja); Petra Wängler, Eva Hierlmeier (Koordination)
Redaktionsassistenz: Monika Lewandowski (222)

Fotografie: Jens Jancke

Titelgestaltung: Heinz Rauner Grafik-Design

Layout: Leo Eder (Ltg.),
Sigrid Kowalewski (Cheflayouterin)
Rolf Raß, Katja Milles

Auslandsrepräsentation:
Schweiz: Markt & Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug,
Tel. (042) 415656, Telex: 862329 mut ch
USA: M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; Tel. 415-366-3600, Telex 752-351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlags AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Produktionsleitung: Klaus Buck (180)

Anzeigenverkauf: Britta Fiebig (211)

Anzeigenverwaltung und Disposition:
Patricia Schiede (172)

Marketingleiter Vertrieb: Hans Hörl (114)

Vertriebsleitung: Helmut Grünfeldt (189)

Verlagsleiter M&T Buchverlag: Günther Frank (212)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Str. 96, 7000 Stuttgart 1, Tel. (0711) 6483-0

Bezugsmöglichkeiten: Leser-Service: Telefon (089) 4613-249. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.

Bezugspreis: Das Einzelheft kostet DM 14,-.

Druck: SOV St.-Otto-Verlag GmbH,
Laubanger 23, 8600 Bamberg

Verantwortlich: Für redaktionellen Teil:
Michael Scharfenberger
Für Anzeigen: Ralph Peter Rauchfuß (126).

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt & Technik Verlag Aktiengesellschaft,
Hans-Pinsel-Straße 2, 8013 Haar bei München,
Telefon (089) 4613-0, Telex 5-22052

Telefon-Durchwahl im Verlag:
Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089/4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.

Aktionäre, die mehr als 25% des Kapitals halten: Otmar Weber, Ingenieur, München; Carl-Franz von Quadt, Betriebswirt, München; Aufsichtsrat: Dr. Robert Dissmann (Vorsitzender), Karl-Heinz Fanselow, Eduard Heilmayr

JETZT AUF SCHNEIDER-COMPUTERN:



DIE PROGRAMM-BIBLIOTHEK FÜR TURBO PASCAL®



TURBO-Lader-Grundpaket

Das TURBO-Lader-Grundmodul ist eine umfangreiche Programm-Bibliothek für den TURBO-Pascal-Programmierer. Sie umfaßt zahlreiche ausführlich dokumentierte Prozeduren und Funktionen, die der Profi zur schnellen Lösung seiner Programmieraufgaben verwenden kann und dem Einsteiger das Erlernen der Pascal-Programmierung erleichtern. Das Grundpaket TURBO-Lader bietet ein breitgefächertes Spektrum von Routinen, beginnend bei Bitmanipulation über optimierte Sortierverfahren bis hin zur Anwendung von Splinefunktionen, Fouriertransformation und Regressionsanalyse. Des weiteren Disketten-Routinen zum Lesen eines Inhaltsverzeichnisses oder zum Lesen und Schreiben einzelner Sektoren, Routinen zur Datenüberprüfung, ein Spooler mit Steuerroutinen, erweiterte Stringverarbeitung und vieles mehr. Alle Routinen werden im kommentierten Quellcode für den TURBO-Pascal-Compiler ausgeliefert.

Das TURBO-Lader-Grundpaket erfordert den TURBO-Pascal-Compiler. Es ist lieferbar auf 3″- und 5 1/4″-Disketten und lauffähig auf dem Schneider CPC 464, CPC 664, CPC 6128 und Joyce.

3″-Disk. Best.-Nr. MS 413
5 1/4″-Disk. Best.-Nr. MS 415

DM 138,–/sFr. 125,–/öS 1380,–*

*inkl. MwSt., unverbindliche Preisempfehlung.

Markt&Technik
Schneider CPC-Software
Turbo Lader Business
Die Programm-Bibliothek für Turbo Pascal
Ein komfortabler Bildschirm-Maskengenerator und eine professionelle Dateiverwaltung in Turbo Pascal Source Code.
3″ Schneider-Format

TURBO-Lader Business

TURBO-Lader Business umfaßt einen komfortablen Bildschirm-Maskengenerator und eine professionelle Dateiverwaltung. Der Maskengenerator gibt dem Pascal-Programmierer ein Werkzeug zur einfachen Bearbeitung von Bildschirm-Masken in die Hand. Eine Maske kann beliebig viele Textfelder, bis zu 128 Eingabe- und 128 Ausgabefelder enthalten. Eingabefelder können auf komfortable Art editiert und auf Gültigkeit überprüft werden. Das Dateiverwaltungsmodul unterstützt die Programmierung von Datenbankanwendungen und Stammdatenverwaltungen. Es besteht aus einer komfortablen Datensatz- und Indexverwaltung mit mehreren Schlüsseln und Index-Dateien, die einen sekundenschnellen Zugriff auf beliebige Daten ermöglicht. Mit diesen beiden Modulen stehen dem Anwendungsprogrammierer zwei professionelle Werkzeuge zur zeit- und kostensparenden Erstellung kommerzieller Anwendungen zur Verfügung. Alle Routinen werden im kommentierten Quellcode für den TURBO-Pascal-Compiler ausgeliefert.

TURBO-Lader Business erfordert den TURBO-Pascal-Compiler und das TURBO-Lader-Grundpaket. Es ist lieferbar auf 3″- und 5 1/4″-Disketten und lauffähig auf dem Schneider CPC 464, CPC 664, CPC 6128 und Joyce.

3″-Disk. Best.-Nr. MS 423
5 1/4″-Disk. Best.-Nr. MS 425

DM 148,–/sFr. 132,–/öS 1480,–*

Markt&Technik
Schneider CPC-Software
Turbo Lader Science
Die Programm-Bibliothek für Turbo Pascal
Technisch-wissenschaftliche Funktionen und professionelle statistische Methoden für die Bereiche Medizin, Betriebs- und Volkswirtschaft, Technik und Naturwissenschaften in Turbo Pascal Source Code.
3″ Schneider-Format

TURBO-Lader Science

TURBO-Lader Science ist eine Sammlung technisch/wissenschaftlicher Funktionen und professioneller statistischer Verfahren für die Bereiche Medizin, Betriebs- und Volkswirtschaft, Technik und Naturwissenschaften. Das Modul enthält alle arithmetischen Operationen zur Verarbeitung komplexer Variablen inklusive der Umrechnung der Darstellung und die wichtigsten komplexen Funktionen wie Potenz, Wurzel, trigonometrische, transzendente und exponentielle Funktionen. Darüber hinaus ist ein vollständiges Paket zur Verarbeitung komplexer Matrizen und Vektoren enthalten. Der Statistikteil ist ein praktisches und direkt verwendbares Werkzeug zur computerunterstützten, effektiven Datenanalyse. Er umfaßt eine Vielzahl statistischer Funktionen mit den Schwerpunkten Regression und Korrelation, deskriptive Statistik, Faktoranalyse und Testverfahren. Alle Routinen werden im kommentierten Quellcode für den TURBO-Pascal-Compiler ausgeliefert.

TURBO-Lader Science erfordert den TURBO-Pascal-Compiler und das TURBO-Lader-Grundpaket. Es ist lieferbar auf 3″- und 5 1/4″-Disketten und lauffähig auf dem Schneider CPC 464, CPC 664, CPC 6128 und Joyce.

3″-Disk. Best.-Nr. MS 433
5 1/4″-Disk. Best.-Nr. MS 435

DM 189,–/sFr. 169,–/öS 1890,–*

Übrigens können Sie auch alle TURBO-Pascal-Produkte für Schneider CPC 464/664/6128 und Joyce bei Markt & Technik beziehen:

- TURBO Pascal 3.0, Best.-Nr. MS 514 (CPC), Best.-Nr. MS 515 (Joyce) DM 225,72*
- TURBO Pascal 3.0 mit Grafikunterstützung, Best.-Nr. MS 524 (CPC) DM 285,–*
- TURBO Tutor (deutsch), Best.-Nr. MS 534 (CPC), Best.-Nr. MS 535 (Joyce) DM 104,86*
- TURBO Tutor (englisch), Best.-Nr. MS 544 (CPC), Best.-Nr. MS 545 (Joyce) DM 104,86*
- TURBO Graphix Toolbox, Best.-Nr. MS 564 (CPC) DM 225,72*
- TURBO Toolbox, Best.-Nr. MS 554 (CPC), Best.-Nr. MS 555 (Joyce), DM 225,72*

TURBO-Pascal® ist ein Warenzeichen der Borland Inc., USA. TURBO-Lader, TURBO-Lader Business und TURBO-Lader Science sind Warenzeichen der Fa. Lauer & Wallnitz.

Diese Markt & Technik-Softwareprodukte erhalten Sie in den Fachabteilungen der Kaufhäuser und in Computershops.

Wenn Sie direkt beim Markt & Technik Verlag bestellen wollen:
Nur gegen Vorauskasse, Verrechnungsscheck oder mit der eingedruckten Zahlkarte.



Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Bestellungen im Ausland bitte an untenstehende Adressen:

Schweiz: Markt & Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042/41 56 56

Österreich: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Str. 24, A-1091 Wien, Tel. 0222/48 15 38-0